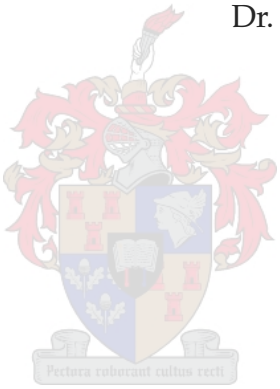
STELLENBOSCH UNIVERSITY

MASTER THESIS

Projected Naïve Bayes

Author:
Michail C. MELONAS

Supervisor:
Dr. David P. HOFMEYR



A thesis submitted in partial fulfillment of the requirements for the degree of Master of Commerce (Mathematical Statistics)

in the

Department of Statistics and Actuarial Science

March 2020

Declaration of Authorship

I, Michail C. MELONAS, declare that this thesis titled, "Projected Naïve Bayes", and the work presented in it are my own. I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Signed:

Date:

"A society that puts equality before freedom will get neither. A society that puts freedom before equality will get a high degree of both."

Milton Friedman

Abstract

Projected Naïve Bayes

Michail C. Melonas

Dissertation: MCom

October, 2019

Naïve Bayes is a well-known statistical model that is recognised by the Institute of Electrical and Electronics Engineers (IEEE) as being among the top ten data mining algorithms. It performs classification by making the strong assumption of class conditional mutual statistical independence. Although this assumption is unlikely to be an accurate representation of the true statistical dependencies, naïve Bayes nevertheless delivers accurate classification in many domains. This success can be related to that of linear regression providing reliable estimation in problems where exact linearity is not realistic. There is a rich body of literature on the topic of improving naïve Bayes. This dissertation is concerned with doing so via a projection matrix that provides an alternative representation for the data of interest. We introduce Projected Gaussian naïve Bayes and Projected Kernel naïve Bayes as naïve-Bayes-type classifiers that respectively relies on Gaussianity and kernel density estimation. The proposed method extends the flexibility of the standard naïve Bayes. The approach maintains the simplicity and efficiency of naïve Bayes while improving its accuracy. Our method is shown to be competitive with several popular classifiers on real-world data. In particular, our method's classification accuracy is compared to that of linear- and quadratic discriminant analysis, the support vector machine and the random forest. There is a close connection between our proposal and the application of naïve Bayes to a class conditionally conducted independent component analysis. In addition to a classification accuracy improvement, the proposed method also provides a tool for visually representing data in low-dimensional space. This visualisation aspect of our method is discussed with respect to the connection to independent component analysis. Our method is shown to give a better visual representation than does linear discriminant analysis on a number of real-world data-sets.

Uittreksel

Projected Naïve Bayes

Michail C. Melonas

Proefskrif: MCom

Oktober, 2019

Naïve Bayes is ’n bekende statistiese model wat deur die Institute of Electrical and Electronics Engineers (IEEE) erken word as een van die top tien data-ontginning algoritmes. Dit voer klassifikasie uit deur die sterk aanname van klasvoorwaardelike onderlinge statistiese onafhanklikheid te maak. Alhoewel hierdie aanname waarskynlik nie ’n akkurate voorstelling van die werklike statistiese afhanklikhede is nie, lewer naïve Bayes nietemin akkurate klassifikasie in baie domeine. Hierdie sukses hou verband met dié van lineêre regressie, wat betroubare beraming gee in probleme waar presiese lineariteit nie realisties is nie. Daar is ’n ryk literatuur rondom die verbetering van naïve Bayes. Hierdie proefskrif handel daaroor via ’n projeksiematriks wat ’n alternatiewe voorstelling bied vir die data van belang. Ons stel “Projected Gaussian naïve Bayes” en “Projected Kernel naïve Bayes” voor as naïve-Bayes-tipe klassifiseerders wat onderskeidelik gebruik maak van die Normaalverdeling en kerndigtheidsberaming. Die voorgestelde metode brei die buigsaamheid van die standaard naïve Bayes uit. Die benadering handhaaf die eenvoud en doeltreffendheid van naïve Bayes terwyl die akkuraatheid daarvan verbeter word. Daar word getoon dat ons metode mededingend is met verskeie gewilde klassifiseerders op natuurlike data. In besonder word die akkuraatheid van die klassifikasie van ons metode vergelyk met dié van lineêre- en kwadratiese diskriminant analiese, die “support vector machine” en die “random forest”. Daar is ’n noue verband tussen ons voorstel en die toepassing van naïve Bayes op ’n voorwaardelike onafhanklike komponentanalise. Benewens die verbetering van die akkuraatheid van klassifikasie, bied die voorgestelde metode ook ’n instrument om data in ’n lae-dimensionele ruimte visueel voor te stel. Hierdie visualiseringsaspek van ons metode word bespreek met betrekking tot die verbinding met onafhanklike komponentanalise. Ons metode word getoon om ’n beter visuele voorstelling te gee as wat lineêre diskriminant analiese op ’n aantal natuurlike datastelle doen.

Acknowledgements

I am sincerely grateful for the opportunity to have worked with Dr. David Hofmeyr. I wish to acknowledge his contribution in being responsible for the idea that underpins this thesis. I am very thankful for his consistently thorough, enthusiastic and speedy feedback during the development and writing of this dissertation. In addition, I formally acknowledge the National Research Foundation for their financial assistance. Opinions expressed, and conclusions arrived at, in this thesis are those of the author and are not necessarily to be attributed to the National Research Foundation.

Contents

List of Abbreviations

BFGS	**B**royden-**F**letcher-**G**oldfarb-**S**hanno
CC	**C**lass **C**onditional
CCICA	**C**lass **C**onditional **I**ndependent **C**omponent **A**nalysis Naïve Bayes
CDF	**C**umulative **D**istribution **F**unction
CLT	**C**entral **L**imit **T**heorem
ICA	**I**ndependent **C**omponent **A**nalysis
IID	**I**ndependent and **I**dentically **D**istributed
KDE	**K**ernel **D**ensity **E**stimation
LDA	**L**inear **D**iscriminant **A**nalysis
ML	**M**aximum **L**ikelihood
NB	**N**aïve **B**ayes
PCA	**P**rincipal **C**ompnent **A**nalysis
PDF	**P**robability **D**ensity **F**unction
PGNB	**P**rojected **G**aussian **N**aïve **B**ayes
PKNB	**P**rojected **K**ernel **N**aïve **B**ayes
QDA	**Q**uadratic **D**iscriminant **A**nalysis
RF	**R**andom **F**orest
SGNB	**S**tandard **G**aussian **N**aïve **B**ayes
SKNB	**S**tandard **K**ernel **N**aïve **B**ayes
SVM	**S**upport **V**ector **M**achine

Dedicated to life, liberty and the pursuit of happiness.

Chapter 1

Introduction: Naïve Bayes

This thesis presents an improvement to the standard naïve Bayes (NB) classifier. Gradient-based learning of a projection matrix delivers a feature representation that significantly improves classification accuracy. The likelihood under NB is utilised which gives an objective function similar to that due to independent component analysis (ICA). The method proposed can be viewed as an attempt to traverse the bias-variance trade-off: additional parameters addresses the issue of biased approximations typical of NB, and provide the freedom that allows for an overall improvement in classification accuracy. This dissertation contributes to the growing literature on the topic of improving NB in ways that maintain its interpretational and computational simplicity. In this introductory chapter, we first consider the standard NB classifier and its historical roots. We also briefly discuss domains where NB has been shown to perform well. Then, we look at NB's theoretical underpinnings and carefully study its assumptions. In support of the distinction between class conditional and unconditional statistical independence, the well-known Simpson's paradox is presented. Finally, a review of existing literature which extends the standard NB followed by a brief outline of the thesis is provided.

1.1 In the Beginning

It is difficult to pinpoint the genesis of the naïve Bayes classifier. In part, this can be attributed to a lack of consensus on the ownership of one of the method's core ingredients: Bayes' theorem. The strongest case against Thomas Bayes himself was made by Stigler (1983) in favour of Nicholas Saunderson. In his paper, Stigler presents a colourful case that introduces Saunderson as an undeservingly little known historical figure, and proposes that he should, at the very least, be considered a contender. Saunderson is interesting for many reasons, but it is worth noting that the man occupied a position at the University of Cambridge once held by Isaac Newton, despite his handicap of blindness. Ironically, during his time as Professor of Mathematics, Saunderson's specialty was optics.

The truth challenged by Stigler is that the well-known formula:

$$\mathrm{P}(A \mid B) = \frac{\mathrm{P}(B \mid A)\mathrm{P}(A)}{\mathrm{P}(B)}, \tag{1.1.1}$$

where A and B represent any suitably defined events, should be attributed to Richard Price's posthumous publication of Bayes' work in 1764. Appropriately, Stigler uses a Bayesian analysis to argue that the odds of the theorem belonging to Saunderson, rather than Bayes himself, is 3 to 1. Irrespective of who was first, Bayes or Saunderson, the above is so fundamental to statistics that Sir Harold Jefferys famously said that it is "to the theory of probability, what is Pythagoras' theorem to geometry" (Debnath & Basu, 2014:16).

The journey from the introduction of (1.1.1) into an explicit classification rule is difficult to track. The earliest example that could be found in the literature was presented by Maron for the RAND Corporation (1961). However, in our opinion, it is very possible that at some point after the surface of the above formula and before 1961, a classification rule that was both naïve in its multivariate treatment (i.e., ignorant of class conditional dependencies – see below) and

reliant on Bayes' theorem, could have been developed and used under the conviction that it was merely an application of (1.1.1), rather than a notable contribution. Hence, it is believable that such a development could have gone unpublished.

Another reason it is difficult to claim an exact inception of NB (beyond the uncertainty of when (1.1.1) first came about) is thus due to its simplicity. The fact that no evidence of an earlier version of the classifier could be found is not conclusive that no such version existed. After all – absence of evidence, is not evidence of absence.

Maron's contribution was an algorithm that indexed documents according to their content. Specifically, he developed a classifier that sorted abstracts extracted from the March, June and September 1959 issues of the *IRE Transactions on Electronic Computers* journal into 32 distinct categories. His proposal relied on the two critical ingredients that constitute what is now known as the NB classifier: Bayes' theorem and class conditional (CC) mutual statistical independence.

1.2 The Standard Classifier

In a classification setting the task is to assign an observation to one of a pre-determined collection of categories. Such an observation is typically pre-processed into, and then characterised by, a vector of measurements. These measurements are the input of a statistical classification rule, and can be thought of as representatives of the raw observation (being either discrete, nominal or continuous quantities). With supervised learning, one has access to a collection of sample observations that are each paired with a corresponding class label. The philosophy of this type of learning is that there is regularity in the sample data that can be discovered, and that once this regularity is established, it can be applied to unlabeled cases – those outside the sample collection that have not already been diagnosed by some domain expert. The idea then is to learn from the sample in the hope that, upon graduation, one has a classifier that is at least better at sorting than would be the case under random guessing.

The nature of a classifier is determined by how it approaches this regularity. In Maron's problem of document indexing, he figured out that the presence of certain words can be used as indicative clues; clues from which a probabilistic inference regarding group, or class, can be made. In his problem then, each document abstract is the raw input, and the conversion of each abstract into a binary-valued vector representing the presence of the identified clue words in any particular such abstract, the pre-processed summary.

1.2.1 Framework and Notation

Throughout this thesis we will assume access to a collection of n pairs, $\{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$. These pairs are assumed to represent independent and identically distributed (IID) realisations from an unknown joint distribution, say $F_{(\mathbf{X},\mathrm{Y})}$, and are stored in matrix form:

$$\underset{n\times p}{\boldsymbol{X}} = \begin{bmatrix} \boldsymbol{x}_1^\top \\ \boldsymbol{x}_2^\top \\ \vdots \\ \boldsymbol{x}_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & x_{12} & \ldots & x_{1p} \\ x_{21} & x_{22} & \ldots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \ldots & x_{np} \end{bmatrix} \tag{1.2.1}$$

and

$$\underset{n\times 1}{\boldsymbol{y}} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}.$$

Each of the p components of the random vector $\mathbf{X}$, e.g. the random variable X_j, will be termed a "predictor", with the entries in the j^{th} column of $\boldsymbol{X}$ treated as independent realisations of X_j. The rows of $\boldsymbol{X}$ represent the pre-processed input patterns, and each corresponding entry

$y_i \in \{1, 2, \ldots, K\}$ its class label. Here, K represents the number of possible groups to which an observation can belong.

The philosophy of naïve Bayes is then as follows: award an observation, say $\boldsymbol{x}$, to the class maximising $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$. This quantity is computed using (1.1.1), and assuming class conditional independence of the predictors, the latter of which implies that the probability density-, or mass, function of the joint distribution of a collection of random variables is equal to the product of the marginal density-, or mass, functions (see the second paragraph of Section 1.4):

$$\begin{aligned} \mathrm{P}\left(\mathrm{Y} = k \mid \boldsymbol{x}\right) &= \frac{f_{\mathbf{X}}\left(\boldsymbol{x} \mid \mathrm{Y} = k\right) \mathrm{P}\left(\mathrm{Y} = k\right)}{f_{\mathbf{X}}\left(\boldsymbol{x}\right)} \\ &\propto \left(\prod_{j=1}^{p} f_j^k(x_j)\right) \pi^k \qquad k = 1, 2, \ldots, K, \end{aligned} \tag{1.2.2}$$

where $\pi^k = \mathrm{P}(\mathrm{Y} = k)$ and where f_j^k denotes either the probability- density function (PDF), or mass function, of the j^{th} predictor conditional on $\mathrm{Y} = k$ i.e., $\mathrm{X}_j \mid \mathrm{Y} = k$.

The quantity $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$ will be termed the "posterior" probability, which represents the probability that Y takes the value k conditional on observing the realisation, $\mathbf{X} = \boldsymbol{x}$ (i.e., that the realisation, $\boldsymbol{x}$, belongs to the k^{th} class). Similarly, π^k is referred to as the "prior" probability, and is considered the probability that $\mathrm{Y} = k$ before observing any information.

The development in (1.2.2) is that we have traded the problem of finding $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$ for $f_{\mathbf{X}}(\boldsymbol{x} \mid \mathrm{Y} = k)$ via Bayes' theorem. Then, we have exchanged this intimidating multivariate quantity for the far easier product of marginals. To achieve the simplification, we have taken $\{\mathrm{X}_j \mid \mathrm{Y} = k : j = 1, 2, \ldots, p\}$ to be independent for all of the classes.

A collection of random variables are statistically mutually independent if they are such that the value of any one subset of them cannot be used to improve our understanding of any other subset. It is worthwhile to consider this concept in the setting of document indexing. Maron treated the presence of the identified clue words to be independent conditional on the document class. For example, the words "Program" and "Code" were treated as statistically unrelated to one-another, as long as the context was a particular class (e.g. "Cybernetics" or "Boolean algebra"). Importantly, assuming CC independence is not equivalent to assuming that "Program" and "Code" are altogether unrelated (this is discussed at length in Section 1.4). In fact, without knowledge of the class to which a document belongs, the presence of one clue word might very well reveal which other words are likely to also be present. Typically, the distribution of both $\mathrm{X}_j \mid \mathrm{Y} = k$ and Y will be unknown, and going about estimating f_j^k and π^k is the topic of Section 1.2.2.

1.2.2 Estimation of Unknown Quantities

A sensible approximation to the unknown prior probability, π^k, is simply taking each class' proportional representation in $\boldsymbol{y}$ (the vector of observed class labels):

$$\hat{\pi}^k = \frac{1}{n} n^k, \tag{1.2.3}$$

where $n^k = \sum_{i=1}^{n} \mathbb{1}(y_i = k)$, and where $\mathbb{1}(\cdot)$ denotes the indicator function. This is a reasonable choice as it can be shown that (1.2.2) is a strongly consistent estimator. That is, $\hat{\pi}^k \xrightarrow{a.s.} \pi^k$ as $n \to \infty$ (John & Langley, 1995:341). In spite of this optimality, some authors (e.g. Bressan and Vitrià (2002)) choose to assume equiprobable priors, i.e. $\hat{\pi}^k = \frac{1}{K} \;\; \forall \; k$, or to choose $\hat{\pi}^k$ via some other means (see El Hindi (2014)). One motivation for moving $\hat{\pi}^k$ away from (1.2.2) could be that there is an inequality in terms of the misclassification cost over the different classes. Thus, a researcher might deliberately trade-off classification accuracy for a reduced chance of making a particular type of error. This kind of approach is typical of spam filtering,

medical diagnosis, weather predictions and fraud screening. With the latter of these examples, one might imagine there being greater cost associated with falsely labeling a fraudulent transaction as being legitimate, than when the reverse takes place.

The estimation of f_j^k will depend on the nature of the conditional distribution of the j^{th} predictor. Typically, a parametric assumption is made. E.g., should this predictor be interpretable as the number of successes of n_j (assumed to be known) independent trials, one could take:

$$X_j \mid Y = k \sim \text{Binomial}(n_j, p_j^k)$$

for $j = 1, 2, \ldots, p$. (In the case of Maron's document classification, each conditional predictor was modeled as a Bernoulli random variable i.e., $n_j = 1$). Looking at:

$$L = \prod_{\ell : y_\ell = k} \binom{n_j}{x_{\ell j}} (p_j^k)^{x_{\ell j}} (1 - p_j^k)^{n_j - x_{\ell j}},$$

the maximum likelihood (ML) estimate for the unknown p_j^k follows as:

$$\hat{p}_j^k = \frac{1}{n^k} \sum_{\ell : y_\ell = k} \frac{x_{\ell j}}{n_j}, \tag{1.2.4}$$

where we can interpret $\hat{p}_j^k$ as the empirical average of the success proportions, $\frac{x_{\ell j}}{n_j}$ (with $x_{\ell j}$ denoting the number of successes). In the event that $x_{\ell j} = 0 \quad \forall \ \ell : y_\ell = k$ (perhaps due to very large K, the number of classes, or a small sample size, n), (1.2.2) will collapse to zero since $\hat{p}_{kj} = 0$. This is not ideal as it ignores the information contributed by the other predictors. As a preventative measure, one could alter the ML estimate slightly, viz. swap (1.2.4) for:

$$\hat{p}_j^k = \frac{1}{n^k + \epsilon} \left(\sum_{\ell : y_\ell = k} \frac{x_{\ell j}}{n_j} + \epsilon \right), \tag{1.2.5}$$

where $\epsilon > 0$ is some small amount. This is known as the Laplace adjustment (Bai & Nie, 2004:2) – the benefit of which is clear in the document classification setup. Imagine some clue word does not show up in any of the sample cases of a particular class. Then, should this word be present in a future document, regardless of how convincing the other clue words (i.e. $\hat{p}_t^k$ for $t \neq j$ might all be close to 1), (1.2.3) will be zero and so the document's estimated probability of belonging to that class will be zero. Thus, the Laplace adjustment can very well prevent the disposal of key discriminating information.

Alternatively, a predictor might be interpretable as the number of independent events that take place within some time interval. Then, it could make sense to assume

$$X_j \mid Y = k \sim \text{Poisson}(\lambda_j^k),$$

where $E(X_j \mid Y = k) = \text{var}(X_j \mid Y = k) = \lambda_j^k$. Or, should $X_j \mid Y = k$ be continuous and defined on $\mathbb{R}$,

$$X_j \mid Y = k \sim \text{Normal}(\mu_j^k, \sigma_{jj}^k), \tag{1.2.6}$$

where $E(X_j \mid Y = k) = \mu_j^k$ and $\text{var}(X_j \mid Y = k) = \sigma_{jj}^k$, is commonly taken. The ML estimates can then be shown to be:

$$\hat{\lambda}_j^k = \frac{1}{n^k} \sum_{\ell : y_\ell = k} x_{\ell j}$$

and

$$\hat{\mu}_j^k = \frac{1}{n^k} \sum_{\ell : y_\ell = k} x_{\ell j} \tag{1.2.7}$$

$$\hat{\sigma}_{ij}^k = \frac{1}{n^k} \sum_{\ell : y_\ell = k} (x_{\ell i} - \hat{\mu}_i^k)(x_{\ell j} - \hat{\mu}_j^k). \tag{1.2.8}$$

Similar to (1.2.4), the Laplace adjusted counterparts for these ML quantities can also be given.

From Chapter 2 onwards we shall focus on the case where $\mathbf{X}$ is a vector of continuous random variables each being defined on $\mathbb{R}$. Thus, unless otherwise stated, we assume each predictor, X_j, to be a continuous quantity. The incorporation of discrete and nominal valued predictors into our framework will be briefly addressed in Section 5.1. The approach presented thus far constitutes the standard (or, typical) NB classifier. Next, we shall take a look at why this method can be expected to work, and explore its potential pitfalls. After that, we look at applications where NB has done well, and then discuss in greater detail the assumptions of the model and their implications.

1.3 Properties and Applications

The treatment of $X_j \mid Y = k$ as independent quantities for $j = 1, 2, \ldots, p$ is unlikely to be an accurate representation of the true dependencies. One might wonder then (and reasonably so), what could be the benefit of such an easily falsifiable assumption. Why study a seemingly incorrect method? The short answer to this is simple: it works. Naïve Bayes has a long history of success (Hand & Yu, 2001:386). This success can be related to that of linear regression providing reliable estimation in problems where exact linearity is not realistic.

Following Maron's contribution (1961), naïve Bayes has excelled in document classification tasks. A particular successful application has been found with that of email spam filtering – a mechanism preventing unwanted messages from reaching a user's inbox. (See Metsis, Androutsopoulos and Paliouras (2006) for a study of NB and spam filtering). Another area of success has been in the field of medicine. Hand and Yu (2001:387) claim success of NB in its application to the prediction of diseases of the heart, thyroid, liver and abdomen, as well as to that of cancer.

Naiveté: The Unsophisticated Upper Hand

NB can be assessed along the lines of two fundamental themes in statistics: the bias-variance trade-off, and the curse of dimensionality. In this light, we try to account for the classifier's success. Modeling the unknown multivariate density, $f_{\mathbf{X}}(\boldsymbol{x} \mid Y = k)$, as the product of marginals, i.e. taking $f_{\mathbf{X}}(\boldsymbol{x} \mid Y = k) = \prod_{j=1}^p f_j^k(x_j)$, ignores the effect of interactions among the predictors within each class. In most cases, this implies that the approximated density will suffer from a large bias. However, this ignorance also means fewer parameters – fewer unknown quantities that need to be estimated from the sample data. Consider, for example, the case of (1.2.6). Here, we have assumed that the covariance matrix:

$$\boldsymbol{\Sigma}^k = \begin{bmatrix} \sigma_{11}^k & \sigma_{12}^k & \ldots & \sigma_{1p}^k \\ \sigma_{21}^k & \sigma_{22}^k & \ldots & \sigma_{2p}^k \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{p1}^k & \sigma_{p2}^k & \ldots & \sigma_{pp}^k \end{bmatrix}, \tag{1.3.1}$$

where $\sigma_{ij}^k = \operatorname{cov}(X_i, X_j \mid Y = k)$, is diagonal (i.e. $\sigma_{ij}^k = 0 \quad \forall\ i \neq j$). This follows from independence being a sufficient condition for zero covariance.

The consequence of the above is that the number of parameters involved grows linearly, rather than quadratically, in p. Now, since we know that in high dimensions the training samples sparsely populate the input space (Hastie, Tibshirani & Friedman, 2008:23), and we also know that the more parameters involved, the more variable the estimate of $f(\boldsymbol{x} \mid Y = k)$, this linearity becomes very useful: the curse of dimensionality implies that as p increases, the benefit of the reduction in variance increases.

What, then, can be said about the bias? As we might expect, the reduction in variance described above must come at a cost. Hand and Yu (2001:388) attempt to offer an explanation by looking at the odds ratio in the extreme case where predictors, conditional on $\mathrm{Y} = k$, are perfectly correlated:

$$\begin{aligned}\frac{\mathrm{P}(\mathrm{Y}=k_1 \mid \boldsymbol{x})}{\mathrm{P}(\mathrm{Y}=k_2 \mid \boldsymbol{x})} =& \frac{f_{\mathbf{X}}(\boldsymbol{x} \mid \mathrm{Y}=k_1)\pi^{k_1}}{f_{\mathbf{X}}(\boldsymbol{x} \mid \mathrm{Y}=k_2)\pi^{k_2}} \\ =& \frac{f_{\mathrm{X}}^{k_1}(x)}{f_{\mathrm{X}}^{k_2}(x)} \frac{\pi^{k_1}}{\pi^{k_2}},\end{aligned}$$

which follows from perfect correlation implying that the components of $\mathbf{X}$ are all identical, and so evaluating the joint distribution is equal to the evaluation of the distribution of any of the components (denoted as X). Now, having assumed independence, the odds ratio would be approached as:

$$\begin{aligned}\frac{\mathrm{P}(\mathrm{Y}=k_1 \mid \boldsymbol{x})}{\mathrm{P}(\mathrm{Y}=k_2 \mid \boldsymbol{x})} =& \frac{\left(\prod_{j=1}^{p} f_{\mathrm{X}}^{k_1}(x_j)\right)\pi^{k_1}}{\left(\prod_{j=1}^{p} f_{\mathrm{X}}^{k_2}(x_j)\right)\pi^{k_2}} \\ =& \left(\frac{f_{\mathrm{X}}^{k_1}(x)}{f_{\mathrm{X}}^{k_2}(x)}\right)^{p} \frac{\pi^{k_1}}{\pi^{k_2}},\end{aligned} \tag{1.3.2}$$

since it must be that $x_1 = x_2 = \ldots = x_p$.

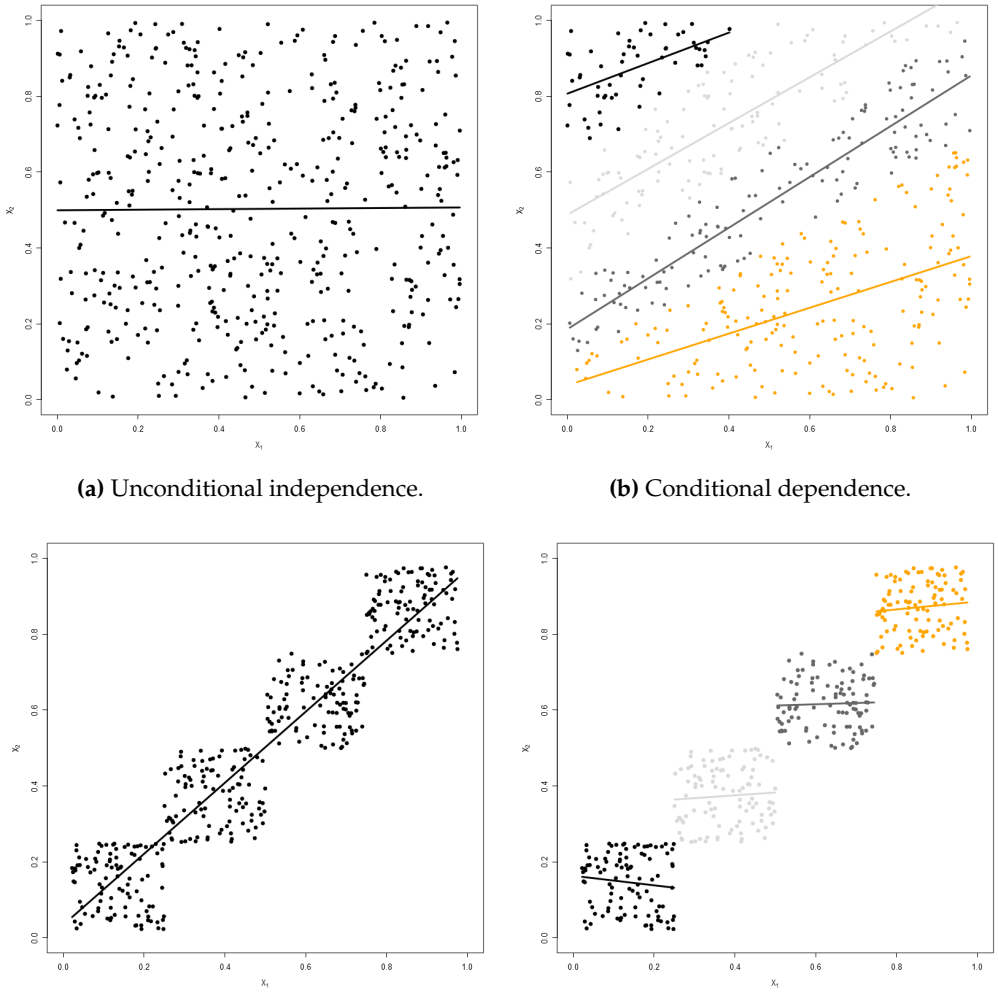
A comparison of the above suggests that if $\frac{f_{\mathrm{X}}^{k_1}(x)}{f_{\mathrm{X}}^{k_2}(x)} < 1$, then (1.3.2) will tend to underestimate the odds. Similarly, $\frac{f_{\mathrm{X}}^{k_1}(x)}{f_{\mathrm{X}}^{k_2}(x)} > 1$ means that $\frac{\mathrm{P}(\mathrm{Y}=k_1|\boldsymbol{x})}{\mathrm{P}(\mathrm{Y}=k_2|\boldsymbol{x})}$ will tend to be overestimated. Furthermore, as p grows, the extent of this bias will increase. (Note, however, that the above is the most extreme case, and that in general the bias will be far less).

We can therefore expect that as the dimension of the problem increases, both the benefit and downside offered by the independence assumption will get exaggerated. The issue, then, is whether the additional bias is justified by the reduction in variance. One reason this could in fact be the case is that when classification is our aim, slight bias of the posterior probabilities might not be an atrocity: following from (1.2.2), as long as

$$\begin{aligned}\underset{k=1,2,\ldots,K}{\arg\max}\ \mathrm{P}\left(\mathrm{Y}=k \mid \boldsymbol{x}\right) =& \underset{k=1,2,\ldots,K}{\arg\max}\ \hat{\mathrm{P}}\left(\mathrm{Y}=k \mid \boldsymbol{x}\right) \\ =& \underset{k=1,2,\ldots,K}{\arg\max} \left(\prod_{j=1}^{p} \hat{f}_j^{k}(x_j)\right) \hat{\pi}^{k},\end{aligned} \tag{1.3.3}$$

the actual values found for $\hat{\mathrm{P}}\left(\mathrm{Y}=k \mid \boldsymbol{x}\right)$, the estimated posterior probability, are unimportant. Thus, the only condition necessary for success (Rish, 2001:1) is that the both the estimated and actual probability terms agree on the class that is most likely. (This being said, we still expect better estimates of $\mathrm{P}(\mathrm{Y}=k \mid \boldsymbol{x})$ to translate into improved classification accuracy).

On a final note, the quality of NB will be dependent on the appropriateness of the parametric assumption made. E.g., if Normality is a reasonable treatment for the distribution of $\mathrm{X}_j \mid \mathrm{Y}=k$, for $j=1,2,\ldots,p$ and $k=1,2,\ldots,K$, then the NB classifier will enjoy a comparative advantage over a method which does not exploit this structure. However, if Normality is inappropriate, then we would also expect a similar comparative disadvantage. The benefit of estimating f_j^k non-parametrically (see Section 1.5 below) was proposed by John and Langley (1995), and this delivered notable improvements on several standard data-sets.



(a) Unconditional independence.

(b) Conditional dependence.

(c) Unconditional dependence.

(d) Conditional independence.

Figure 1.1: Simpson's Paradox.

1.4 Simpson's Paradox

In the naïve Bayes context, a point that deserves to be made is that of unconditional versus conditional independence. Specifically, how either does not give us the other. In support of this, Simpson's paradox is (informally) stated: the phenomenon whereby the overall account differs greatly with what happens at the group level.

It was noted in Section 1.2.1 that a collection of random variables, say $X_1, X_2, \ldots, X_p$, are considered statistically mutually independent if these variables are such that the value of any one subset of them cannot be used to improve our understanding of any other subset. Formally, a sufficient condition for independence is when the density- (or mass) function of the joint distribution of the variables, denoted by $f_{\mathbf{X}}$, can be factorised into the product of the marginal densities (or mass functions) of the variables, denoted f_{X_j} for $j = 1, 2, \ldots, p$:

$$f_{\mathbf{X}} = \prod_{j=1}^{p} f_{X_j}.$$

When evaluating the strength of the NB assumption in the set of input patterns, this is something that should be kept in mind, and is what the plots in Figure 1.1 aim to stress. In

these figures, we have fitted the relevant linear regression lines to demonstrate the relationship level being evaluated (not to be confused with implying that correlation and independence are one and the same). Treating the left and right panels as pairs, we visually confirm that a lack of dependence on a global level does not guarantee independence at the group level. Similarly, a global dependence does not imply that the same (or any) relationship holds at the group level.

This relation between the conditional and the unconditional can also be motivated mathematically. Suppose that we have a random vector which is distributed according to a mixture of Gaussians:

$$\mathbf{X} \sim F_{\mathbf{X}}$$

where

$$f_{\mathbf{X}}(\boldsymbol{x}) = \sum_{k=1}^{K} f_{\mathbf{X}|\mathrm{Y}=k}(\boldsymbol{x} \mid \mathrm{Y} = k)\pi^k$$

and $\mathbf{X} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\right)$. Then, we can use this setting to provide a more formal algebraic motivation against the hypotheses that conditional independence guarantees the unconditional kind in a way which takes advantage of the property that for a Normally distributed random vector, zero covariance among its components is equivalent to those components being independent (which was also mentioned with the discussion in Section 1.3.1). This is used together with the general property that non-zero covariance implies dependence.

Now, briefly emphasising our notation, note that:

$$\boldsymbol{\mu}^k = \mathrm{E}(\mathbf{X} \mid \mathrm{Y} = k) = \begin{bmatrix} \mu_1^k \\ \mu_2^k \\ \vdots \\ \mu_p^k \end{bmatrix} \tag{1.4.1}$$

is used to represent the vector of means of the random vector, $\mathbf{X}$, conditional on $\mathrm{Y} = k$, and that $\boldsymbol{\Sigma}^k = \mathrm{cov}(\mathbf{X} \mid \mathrm{Y} = k)$, denotes its covariance matrix. The latter is of the form given in (1.3.1). Our illustration will require a comparison of the conditional and unconditional character of $\mathbf{X}$. For this reason, it is necessary to consider:

$$\begin{aligned} \sigma_{ij} = \mathrm{cov}(\mathrm{X}_i, \mathrm{X}_j) =& \mathrm{E}(\mathrm{X}_i\mathrm{X}_j) - \mathrm{E}(\mathrm{X}_i)\mathrm{E}(\mathrm{X}_j) \\ =& \sum_{k=1}^{K} \mathrm{E}(\mathrm{X}_i\mathrm{X}_j \mid \mathrm{Y} = k)\pi^k \\ & - \left(\sum_{k=1}^{K} \mathrm{E}(\mathrm{X}_i \mid \mathrm{Y} = k)\pi^k\right)\left(\sum_{k=1}^{K} \mathrm{E}(\mathrm{X}_j \mid \mathrm{Y} = k)\pi^k\right) \\ =& \sum_{k=1}^{K} \left(\sigma_{ij}^k + \mu_i^k\mu_j^k\right)\pi^k - \left(\sum_{k=1}^{K} \mu_i^k\pi^k\right)\left(\sum_{k=1}^{K} \mu_j^k\pi^k\right). \end{aligned} \tag{1.4.2}$$

Using the above as our frame of reference, it follows that conditional independence (among all components of $\mathbf{X}$ and over all classes) is achieved when $\sigma_{ij}^k = 0 \quad \forall\ i \neq j$ for $k = 1, 2, \ldots, K$, and that unconditional (or, global) dependence is the case when $\exists$ pair (i, j) with $i \neq j$ such that $\sigma_{ij} \neq 0$. Supposing that $K = p = 2$ and that $\pi^1 = \pi^2 = \frac{1}{2}$, first consider the special case where: $\boldsymbol{\mu}^1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$, $\boldsymbol{\mu}^2 = \begin{bmatrix} 3 \\ 4 \end{bmatrix}$, $\boldsymbol{\Sigma}^1 = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$ and $\boldsymbol{\Sigma}^2 = \begin{bmatrix} 3 & 0 \\ 0 & 4 \end{bmatrix}$. Here, the requirements of conditional independence are clearly met. However, using (1.4.2), we find that:

$$\begin{aligned} \sigma_{12} =& \left(\frac{1}{2}(0+2) + \frac{1}{2}(0+12)\right) - \left(\frac{1}{2}(1+3)\frac{1}{2}(2+4)\right) \\ =& 1 \neq 0, \end{aligned}$$

and so we have X_1 and X_2 are dependent.

To motivate that the reverse also fails i.e., that unconditional independence does not promise conditional independence, we invoke the setting of an unbiased coin toss. Here, we will assume that the random vector $\mathbf{X}$ consists of the independent components, X_1 and X_2, where:

$$X_i \sim \text{Bernoulli}\left(\frac{1}{2}\right) \qquad i = 1, 2.$$

Then, in keeping with our framework, we construct class labels according to the sum of X_1 and X_2:

$$Y = \begin{cases} 1, & \text{if } X_1 + X_2 = 1 \\ 2, & \text{otherwise} \end{cases}$$

Using the fact that the joint probability mass function of independent random variables can be written as the product of the marginal mass functions, the following simple example proves that one can have variables to be unconditionally independent, yet conditionally dependent:

$$\begin{aligned} P(X_1 = 1, X_2 = 1) =& \frac{1}{4} \\ =& P(X_1 = 1)P(X_2 = 1) \\ =& \left(\frac{1}{2}\right)^2 \\ =& \frac{1}{4} \end{aligned}$$

while

$$\begin{aligned} P(X_1 = 1, X_2 = 1 \mid Y = 1) =& 0 \\ \neq& P(X_1 = 1 \mid Y = 1)P(X_2 = 1 \mid Y = 1) \\ =& \left(\frac{1}{2}\right)^2 \\ =& \frac{1}{4}. \end{aligned}$$

The lesson here is that neither condition is stronger than the other (in the sense as presented above). Therefore, the temptation to conflate the two types of independence should be avoided. The relevance of this is further emphasised in the discussion of independent component analysis and its application to NB in Chapters 2 and 3.

1.5 Review: Existing Literature on Extending Standard Naïve Bayes

This thesis is not the only attempt at improving upon the standard naïve Bayes. In fact, many efforts that sought to better the classifier while holding on to both its simplicity and efficiency have been made before. Hoare (2007) composed an extensive overview of such endeavors. She focused on 37 contributions (deemed as worthy), which were, according to character, subdivided into five main groups: tree structures, feature selection methods, space transformation methods, Bayesian networks and joint feature approaches.

It appears that the improvements made to NB have drawn from many different areas of statistics. E.g., Langley and Sage (1994) proposed an approach that applies variable selection before deployment of the NB model. Their idea is that a removal of highly correlated predictors would address its biasing effect, and so improve classification accuracy. The authors searched for features via greedy forward (or backward) stepwise selection.

Another effort is Kohavi's (1996) introduction of a NB hybrid approach. This contribution involved building a decision-tree as per recursive binary partitioning, delivering a collection of non-overlapping input regions. Then, rather than accepting a majority vote for the label of a particular region, a NB model is fitted to each leaf. In effect, this then delivers a collection of NB classifiers. At the time of classification, the member belonging to the region allocated by the tree is invoked as the relevant classifier.

Tsymbal and Puuronen (2003) proposed an ensemble method that involves training a set of NB classifiers. As the standard NB is a low variance technique, and since an ensemble improves things by combining many high variance classifiers, the authors trained each of the ensemble constituents on a randomly selected subset of the available features. In this way, classifiers that are diverse in their predictions can be obtained, and so can benefit from the variance reduction offered by aggregation.

Other efforts include combining NB with boosting, nearest neighbour methods and clustering. The latter of which was introduced by Vilalta and Rish (2003). The clustering reorganises the examples from each class into more closely aligned segments. In this way the number of classes is artificially increased (with multiple clusters corresponding to a single class label). This is done in an effort to avoid the problems that could arise when a class-conditioned predictor has a multi-modal, or heavily skewed, distribution.

Hoare's work highlights that there is no shortage of attempts to refine NB, and so it seems that the art of improving NB is one that appeals to a broad body of researchers. Among the more recent efforts include El Hindi (2014) who presented a fine tuning approach that augments NB with an additional training phase in which probability estimates for misclassified training examples are revised. Another is that of Jiang, Zhang, Li and Wu (2019) who proposed a feature weighting filter that favours the contribution of certain predictors based on a correlation-inspired measure of importance.

Another effort, already alluded to in Section 1.3.1, is the work of John and Langley (1995). These authors' contribution focused on the case of dealing with continuous input features, i.e. where $X_j \mid Y = k$ has a continuous character for $j = 1, 2, \ldots, p$. Their idea was to abandon the typical assumption of Normality in such cases in favour of nonparametric density estimation (see Section 2.1). Assuming Gaussianity (i.e., (1.2.5)) is typically a reasonable thing to do – not because the assumption is particularly valid, but due to its simplicity and low variance implications. As with the discussion of (1.3.3): slight bias of posterior probability estimates is not necessarily an atrocity in the NB context, and so enforcing an inappropriate assumption might have an overall beneficial impact.

However, there are certainly domains in which a more flexible treatment of $X_j \mid Y = k$ can have beneficial consequences on classification accuracy. John and Langley's extension of NB using kernel density estimation (KDE) found improvement over assuming Gaussianity on a number of data-sets in the UCI Repository (Dua & Graff, 2019). One such instance of significant improvement, to which our method is also successfully applied, is the Vehicle Silhouettes data-set (see Table 4.4). Furthermore, comparing the performance of the Gaussian NB with that of the KDE version over the size of training set data, it was established that should the Normality of $\{X_j \mid Y = k : j = 1, 2, \ldots, p\}$ be valid, the two approaches have identical asymptotic performance. However, the Gaussian NB reaches this performance faster. In cases where the Normality assumption did not hold, the Gaussian NB never reached the optimal Bayes error rate whereas the more flexible version of NB making use of KDE to approach the class conditional densities of the predictors, $X_j \mid Y = k$, did in fact. This suggests that the abundance of the training data should be factored into the treatment of the predictors.

1.6 Overview of this Dissertation

The hope of this thesis is to leave the reader thinking of naïve Bayes as not only a benchmark of comparison to more sophisticated methods, but rather as a serious contender: one that, when equipped with the proposed additional tools, remains simple in its motivation, yet competes with the state-of-the-art.

The remainder of this thesis is structured as follows:

- **Chapter 2**: In this chapter we give a literature review of the tools necessary to describe and motivate our own NB contribution. We look here into nonparametric density estimation, which is supported by a discussion on its computational feasibility. Then, we introduce the information theoretic concepts of entropy and mutual information, which leads into the discussion of independent component analysis. Finally, a brief background of the optimisation literature relevant to this thesis is presented.

- **Chapter 3**: Here our proposed improvement to the standard NB classifier is treated in detail. Our idea is to learn an alternative representation of the data via a projection that replaces the original inputs with linear mixtures. The proposed alteration is motivated via a maximum likelihood argument. Numerical optimisation of the parameters defining the projection operation is done according to the algorithm due to Broyden (1970), Fletcher (1970), Goldfarb (1970) and Shanno (1970). The parametric case of treating the transformed set of features as Gaussian random variables is presented first. Subsequently, the strong parametric assumption is relaxed and the Gaussian treatment is swapped for kernel density estimation. The classifiers developed are named Projected Guassian NB and Projected Kernel NB, respectively.

- **Chapter 4**: This chapter illustrates the practical application of the proposed NB improvement. Both simulated and real-world data-sets are considered, and Projected NB is compared with other nonlinear statistical classifiers. By finding a projection which maximises the likelihood under the NB model, the proposed method also allows visualisations of the data which exposes the discrimination of classes. Attention to this visualisation aspect of the method is paid on the data-sets considered. In addition, the visualisation is compared to that due to linear discriminant analysis.

- **Chapter 5**: This final chapter provides suggestions on future improvements of the method. In particular, an extension that treats non-continuous inputs distinctly is given. The benefits and challenges of amending the proposal to perform automatic variable selection is also discussed.

Chapter 2

Background Material

In this chapter we shall review theoretical concepts that are necessary to both the description and justification of our naïve Bayes improvement proposal. The techniques considered here will have consequences regarding the optimisation and computational feasibility of our proposed method. The two main topics that are dealt with are that of kernel density estimation and independent component analysis. The former of these two is intimately related to our own work on naïve Bayes (that treated in Section 3.3), and the latter will be used to provide an additional motivation for our NB proposal. The final section will present a short discussion of the optimisation theory relevant to our method.

2.1 Kernel Density Estimation

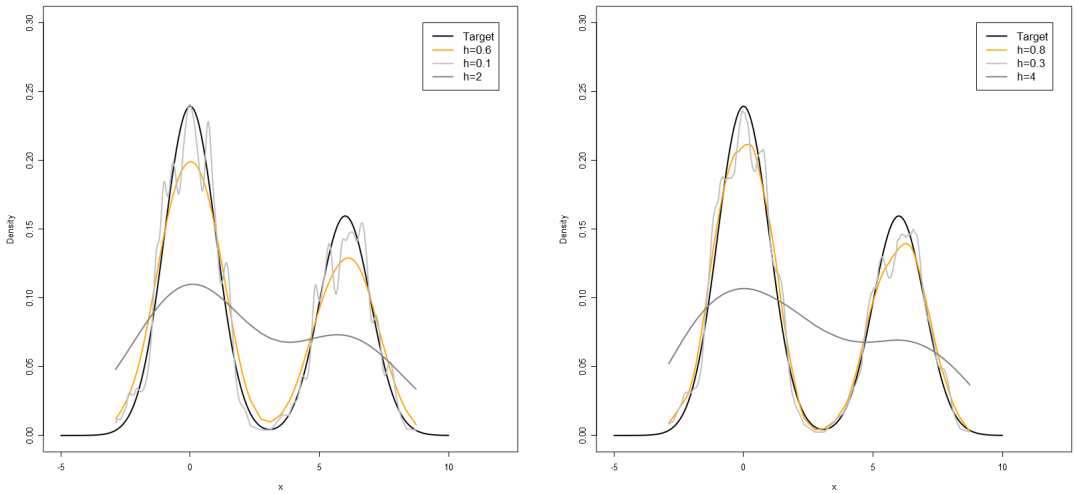
Suppose that we wish to study the distribution of one of the predictors introduced in Section 1.2.1. That is, we are interested in the component, say X_j, of the random vector, $\mathbf{X}$. Typically, the distribution of X_j is not known (although assumed to be continuous and defined on $\mathbb{R}$), and so the quantities related to this random variable need to be approached empirically. Fortunately, we have assumed access to a collection of n observed pairs, $(\boldsymbol{x}_i, y_i)$, of the joint distribution defined by $F_{(\mathbf{X},\mathrm{Y})}$. Following from Section 1.2.1, we have that $\{x_{ij} : i = 1, 2, \ldots, n\}$ is a collection of realisations of independently distributed random variables that each have a distribution identical to that of X_j. One approach to this problem is to make a parametric assumption regarding the distribution of X_j, and then to estimate the relevant parameters using a procedure such as maximum likelihood (see Section 1.2.2). However, it is not necessarily obvious which parametric family constitutes a reasonable assumption. In addition, such methods are of limited flexibility, and it may very well be necessary to investigate random variables which do not conform to the collection of available parametric choices. Kernel density estimation (KDE) provides a means of adaptively estimating density functions without having to make strong assumptions about the structure of the underlying random variable.

Before commencing with this topic, we note that from this point forward we shall drop explicit reference to j, i.e., we shall use X to refer to X_j, and x_i to indicate x_{ij}. This is done for the sake of notational ease, and the convenience of doing so will become clear upon the introduction of notation related to the order statistics (treated in the final part of Section 2.1.2). This section starts with the treatment of KDE, followed by a thorough discussion of computational considerations and then a strategy for addressing this difficulty.

2.1.1 Nonparametric Density Estimation

The natural estimate for the cumulative distribution function (CDF) of X, $F_{\mathrm{X}}(x) = \mathrm{P}(\mathrm{X} \leq x)$, is to assign the empirical proportion in the interval $(-\infty, x]$:

$$\hat{F}_{\mathrm{X}}(x) = \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}(x_i \leq x), \tag{2.1.1}$$



(a) Gaussian kernel. **(b)** Epanechnikov kernel.

Figure 2.1: KDE illustration.

where $\mathbb{1}(\cdot)$ denotes the indicator function. The sensibility of (2.1.1) is confirmed by the Glivenko-Cantelli lemma which promises uniform convergence: $\sup_{x\in\mathbb{R}} \left|F_X(x) - \hat{F}_X(x)\right| \xrightarrow{a.s.} 0$ as $n \to \infty$. The relationship between the CDF and the probability density function of X, f_X, is given by:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)dt.$$

Then, the fundamental theorem of calculus gives that $\frac{d}{dx}F_X(x) = f_X(x)$, and so it would seem that a direct estimate of f_X could simply be taken as: $\frac{d}{dx}\hat{F}_X(x)$. However, since $\hat{F}_X$ is a step function, this would not be possible. Estimating the PDF of X can rather be based on the approximation, $f_X(x) \simeq \frac{1}{2h}\int_{x-h}^{x+h} f_X(x)dx = \frac{1}{2h}\left(F_X(x+h) - F_X(x-h)\right)$ where $h > 0$. Using (2.1.1):

$$\begin{aligned}
\hat{f}_X(x) =& \frac{1}{2h}\left(\hat{F}_X(x+h) - \hat{F}_X(x-h)\right) \\
=& \frac{1}{2nh}\sum_{i=1}^{n} \mathbb{1}(x-h < x_i \leq x+h) \\
=& \frac{1}{2nh}\sum_{i=1}^{n} \mathbb{1}\left(\left(\frac{x-x_i}{h}\right) \in (-1,1]\right) \\
=& \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right), \qquad (2.1.2)
\end{aligned}$$

where here $K(x) = \frac{1}{2}\mathbb{1}(x \in (-1,1])$. In the above, $K(\cdot)$ is termed a "kernel" function, and (2.1.2) the "kernel density estimate" for f_X. This particular choice of K is called the rectangular kernel, however any function that is itself a density function can be taken in its place. Popular choices include the Gaussian kernel, $K(x) = \frac{1}{\sqrt{2\pi}}\exp\left(-\frac{1}{2}x^2\right)$, and the Epanechnikov kernel, $K(x) = \frac{3}{4}(1-x^2)\mathbb{1}(x \in (-1,1]))$.

Figure 2.1 provides an illustration of the KDE. Here, we have simulated realisations from a mixture of two Gaussian distributions (the density function of which is given by the black curve). The estimate (2.1.2) at any particular point is a local average of densities, one for each

observation. In a sense, the kernel function acts as a similarity measure (or, weight function) and decides the contribution of each sample candidate to the evaluation of the density at a particular point.

The quantity h is called the "bandwidth", and its size determines how local the estimate is. Looking at the plots in Figure 2.1 we see that h effectively controls the smoothness of the estimated density: small values correspond to a wiggly curve that is too local (i.e., the resulting estimator assigns too much weight to the observations in close vicinity to the point being evaluated, and discounts the contributions of those further away excessively), and large values of h enacts oversmoothing, moving the estimator toward assigning a near identical density estimate over the entire range of the data. Furthermore, it also appears that the size of h is much more consequential than the choice of kernel function (which is motivated by the fact that both sets of curve estimates under the two different kernel choices in Figure 2.1 were made to be nearly identical with an appropriate choice of bandwidth).

In fact, the choice of h controls the bias-variance trade-off: decreasing h delivers a reduction of bias at the cost of an increased variance, whereas larger choices of h will effect the reverse. The optimal choice of bandwidth (in a mean squared error sense) will depend on the smoothness of the underlying target density. This means that we want to choose the bandwidth as a function of the underlying smoothness, in a way that $h \to 0$ as $n \to \infty$. However, as the distribution of X is not known, this smoothness will be unknown, and so one of the key difficulties with KDE is deciding on a sensible bandwidth estimation approach.

There exist many methods for bandwidth estimation. Heidenreich, Schindler and Sperlich (2013) give a thorough review of the available methods. In particular, cross-validation procedures based on minimising the mean integrated squared error and the so-called "Silverman's rule-of-thumb" are popular choices. Section 3.3.3 describes the choosing of the bandwidth used in the experiments of Chapter 4.

2.1.2 Computationally Efficient Variant

Computationally, KDE is very costly. Evaluating (2.1.2) requires n operations, and so if we were interested in a density estimate for each of the sample points, this effort would grow quadratically. Given the context of our naïve Bayes improvement effort (detailed in Chapter 3), it is paramount that we are able to estimate f_X in a manner that is less computationally dear. This follows from our method requiring repeated evaluation of densities, and their derivatives, over different projections of the data. Hofmeyr (2019a) introduced a computationally efficient approach to the evaluation of univariate kernel sums. Via a smart choice of kernel function, the cost of density (and density derivative) estimation was reduced to being linear in the sample size (discounting the cost of sorting the data).

Class of Kernels Allowing Fast Computation

The first point of order is to notice that (2.1.2) can be viewed as a sum of weighted kernels:

$$\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) w_i, \tag{2.1.3}$$

where $w_i = \frac{1}{nh}$ (with dependence on i stressed here for the sake of generality – in Section 3.3 we consider the case where the w_i are not all equal). The Gaussian kernel, introduced earlier, is widely used. It would therefore be justified to seek an alternative kernel that resembles this trusted choice, while having the added benefit of being able to be manipulated into offering some kind of computational gain. Consider the kernel function given by:

$$K(x) = C_\alpha e^{-d_\alpha |x|} \sum_{k=0}^{\alpha-1} \beta_{\alpha k} |x|^k, \tag{2.1.4}$$

where $\alpha \in \mathbb{N}$, $\beta_{\alpha k} = \binom{\alpha-1}{k}\frac{\Gamma(2\alpha-k-1)}{2^{2\alpha-k-1}}(2\alpha)^{\frac{k}{2}}$, $C_\alpha = \frac{\sqrt{2\alpha}}{(\Gamma(\alpha))^2}$, $d_\alpha = \sqrt{2\alpha}$ and with $\Gamma(\cdot)$ indicating the Gamma function (defined below). It turns out that:

$$\lim_{\alpha\to\infty} K(x) = \frac{1}{2\pi}e^{-\frac{1}{2}x^2}. \tag{2.1.5}$$

I.e., that the Gaussian kernel is a limiting case of (2.1.4). Furthermore, (2.1.3) with K as in (2.1.4) can be manipulated into a form that allows for a reduction in computational cost.

Convergence to the Gaussian Kernel

Consider two independent collections of random variables, say $\{X_i : i = 1,2,\ldots,\alpha\}$ and $\{Y_i : i = 1,2,\ldots,\alpha\}$ where $X_i \sim$ Exponential(1) and $Y_i \sim$ Exponential(1) (with entries in both collections being independent and identically distributed). Then, it is easy to show that $X_i - Y_i$ has a standard Laplace distribution, having density function (see the light grey curve in Figure 2.2):

$$f_{X_i-Y_i}(z) = \frac{1}{2}e^{-|z|}.$$

The Central Limit Theorem (CLT) promises convergence in distribution:

$$\frac{\frac{1}{\alpha}\sum_{i=1}^{\alpha}(X_i-Y_i) - E\left(\frac{1}{\alpha}\sum_{i=1}^{\alpha}(X_i-Y_i)\right)}{\sqrt{\text{Var}\left(\frac{1}{\alpha}\sum_{i=1}^{\alpha}(X_i-Y_i)\right)}} = \frac{1}{\sqrt{2\alpha}}\sum_{i=1}^{\alpha}(X_i-Y_i) \xrightarrow{d} N(0,1) \text{ as } \alpha\to\infty,$$

where $N(0,1)$ represents a standard Gaussian random variable. We show that this convergence implies that the density of $\frac{1}{\sqrt{2\alpha}}\sum_{i=1}^{\alpha}(X_i-Y_i)$ (which is denoted by f_α) converges to the Gaussian kernel, (2.1.5). First, however, we are concerned with the equality of (2.1.4) and f_α (see below). Keeping in mind that the sum of independent exponential random variables has a gamma distribution, it follows that the density of a constant multiple of the sum of IID Laplace random variables can be found via the convolution of two Gamma distributed random variables.

Letting $X = \sum_{i=1}^{\alpha} X_i \sim \text{Gamma}(\alpha,1)$ and $Y = \sum_{i=1}^{\alpha} Y_i \sim \text{Gamma}(\alpha,1)$, and noting $\Gamma(k) = \int_0^\infty x^{k-1}e^{-x}dx$, with some help from Zare (2011) we see that:

$$\begin{aligned}
f_{\sum_{i=1}^{\alpha}(X_i-Y_i)}(z) &= (f_X * f_{-Y})(z)\\
&= \int_{-\infty}^{\infty} f_{-Y}(z-x)f_X(x)dx\\
&= \int_{-\infty}^{\infty} f_Y(x-z)f_X(x)dx\\
&= \begin{cases} \int_0^\infty f_Y(x-z)f_X(x)dx, & \text{if } z \le 0\\ \int_z^\infty f_Y(x-z)f_X(x)dx, & \text{if } z > 0 \end{cases}\\
&= \begin{cases} \int_0^\infty f_X(x)f_Y(x-z)dx, & \text{if } z \le 0\\ \int_0^\infty f_Y(y)f_X(y+z)dy, & \text{if } z > 0 \end{cases}
\end{aligned}$$

$$
\begin{aligned}
&= \begin{cases} \frac{1}{(\Gamma(\alpha))^2} \int_0^\infty e^{-2x+z}(x^2 - zx)^{\alpha-1} dx, & \text{if } z \leq 0 \\ \frac{1}{(\Gamma(\alpha))^2} \int_0^\infty e^{-2y-z}(y^2 + zy)^{\alpha-1} dy, & \text{if } z > 0 \end{cases} \\
&= \frac{1}{(\Gamma(\alpha))^2} \int_0^\infty e^{-2y-|z|}(y^2 + |z|y)^{\alpha-1} dy \\
&= \frac{1}{(\Gamma(\alpha))^2} \int_0^\infty e^{-2y-|z|} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} (y^2)^{(\alpha-1)-k} (|z|y)^k dy \\
&= \frac{1}{(\Gamma(\alpha))^2} e^{-|z|} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} |z|^k \int_0^\infty y^{2(\alpha-1)-k} e^{-2y} dy \\
&= \frac{1}{(\Gamma(\alpha))^2} e^{-|z|} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} \frac{\Gamma(2\alpha-k-1)}{2^{2\alpha-k-1}} |z|^k. \qquad (2.1.6)
\end{aligned}
$$

Finally, it follows that

$$
f_{\frac{1}{\sqrt{2\alpha}}\sum_{i=1}^{\alpha}(X_i - Y_i)}(x) = f_\alpha(x) = \frac{\sqrt{2\alpha}}{(\Gamma(\alpha))^2} e^{-|\sqrt{2\alpha}x|} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} \frac{\Gamma(2\alpha-k-1)}{2^{2\alpha-k-1}} |\sqrt{2\alpha}x|^k,
$$

which is exactly of the form (2.1.4), thereby demonstrating that the density of a constant multiple of the sum of IID Laplace random variables is equal to the kernel function of interest.

The CLT result of convergence in distribution means that for any fixed $x \in \mathbb{R}$ and any small $\epsilon > 0$, $\exists\ \alpha_\epsilon \in \mathbb{N}$ such that:

$$
\begin{aligned}
&|(F_\alpha(x+\epsilon) - F_\alpha(x-\epsilon)) - (\Phi(x+\epsilon) - \Phi(x-\epsilon))| \\
\leq &|F_\alpha(x+\epsilon) - \Phi(x+\epsilon)| + |F_\alpha(x-\epsilon) - \Phi(x-\epsilon)| \\
\leq &\epsilon^2 + \epsilon^2 = 2\epsilon^2 \quad \forall\ \alpha > \alpha_\epsilon, \qquad (2.1.7)
\end{aligned}
$$

which follows from the triangle inequality. Note that, similar to notation introduced earlier, we take F_α to represent the CDF of $\frac{1}{\sqrt{2\alpha}}\sum_{i=1}^{\alpha}(X_i - Y_i)$ and, from the above going forward, make use of the standard notation for the PDF and CDF of a Gaussian distributed random variable (ϕ and Φ, respectively).

Now, first-off, by inspection it follows that both $\lim_{x\to\pm\infty} \phi^{(2)}(x) = 0$ and $\lim_{x\to\pm\infty} f_\alpha^{(2)}(x) = 0$ where the superscript (m) is used to indicate the m^{th} derivative. This fact means that for any $\tau > 0$, $\exists\ m_{1\tau} \in \mathbb{R}$ with $m_{1\tau} > 0$ such that:

$$
|\phi^{(2)}(x) - 0| = |\phi^{(2)}(x)| < \tau \quad \forall\ x > m_{1\tau}
$$

and $\exists\ m_{2\tau} \in \mathbb{R}$ with $m_{2,\tau} < 0$ such that:

$$
|\phi^{(2)}(x) - 0| = |\phi^{(2)}(x)| < \tau \quad \forall\ x < m_{2\tau}.
$$

Letting $m_\tau = \max\{m_{1\tau}, -m_{2\tau}\}$, it must be that $|\phi^{(2)}(x)| < \tau \quad \forall\ x \in \mathbb{R} \setminus [-m_\tau, m_\tau]$. Then, since $[-m_\tau, m_\tau]$ is a compact set, and since any continuous function on a compact set must be bounded, it follows that there must be some $\tau^* > 0$ such that $|\phi^{(2)}(x)| < \tau^* \quad \forall\ x \in [-m_\tau, m_\tau]$. Having $B_1 = \max\{\tau, \tau^*\}$, it follows that $|\phi^{(2)}(x)| < B_1 \quad \forall\ x \in \mathbb{R}$. Thus, $\phi^{(2)}$ is bounded. By the exact same argument, we can show that $f_\alpha^{(2)}(x)$ too is bounded. This means there must exist some $B_2 > 0$ such that $|f_\alpha^{(2)}(x)| < B_2 \quad \forall\ x \in \mathbb{R}$. Therefore, taking $B = \max\{B_1, B_2\}$, we have both $|\phi^{(2)}(x)| < B$ and $|f_\alpha^{(2)}(x)| < B\ \forall\ x \in \mathbb{R}$.

Next, we look at the Taylor series expansion of Φ at $x+\epsilon$ and x. The continuity of Φ means we can make use of an integral term to capture the remainder:

$$\Phi(x+\epsilon) - \Phi(x) = \phi(x)\epsilon + \frac{1}{2!}\phi^{(1)}(x)\epsilon^2 + \frac{1}{2!}\int_x^{x+\epsilon} \phi^{(2)}(t)(x+\epsilon-t)^2 dt.$$

Using this, together with the same expression evaluated at $x-\epsilon$ and x, we get:

$$\begin{aligned}\Phi(x+\epsilon) - \Phi(x-\epsilon) =& 2\epsilon\phi(x) + \frac{1}{2!}\int_x^{x+\epsilon} \phi^{(2)}(t)(x+\epsilon-t)^2 dt - \frac{1}{2!}\int_x^{x-\epsilon} \phi^{(2)}(t)(x-\epsilon-t)^2 dt \\ =& 2\epsilon\phi(x) + \frac{1}{2!}\int_x^{x+\epsilon} \phi^{(2)}(t)(x+\epsilon-t)^2 dt + \frac{1}{2!}\int_{x-\epsilon}^{x} \phi^{(2)}(t)(x-\epsilon-t)^2 dt.\end{aligned}$$

Then, again relying on the triangle inequality, we can construct the bound:

$$\begin{aligned}&\left|\frac{1}{2!}\int_x^{x+\epsilon} \phi^{(2)}(t)(x+\epsilon-t)^2 dt + \frac{1}{2!}\int_{x-\epsilon}^{x} \phi^{(2)}(t)(x-\epsilon-t)^2 dt\right| \\ \leq& \frac{1}{2!}\left(\int_x^{x+\epsilon} |\phi^{(2)}(t)(x+\epsilon-t)^2| dt + \int_{x-\epsilon}^{x} |\phi^{(2)}(t)(x-\epsilon-t)^2| dt\right) \\ \leq& \frac{\epsilon^2}{2!}\left(\int_x^{x+\epsilon} |\phi^{(2)}(t)| dt + \int_{x-\epsilon}^{x} |\phi^{(2)}(t)| dt\right) \\ =& \frac{\epsilon^2}{2!}\int_{x-\epsilon}^{x+\epsilon} |\phi^{(2)}(t)| dt \\ \leq& \frac{B\epsilon^2}{2!}\int_{x-\epsilon}^{x+\epsilon} dt \\ =& B\epsilon^3. \end{aligned} \tag{2.1.8}$$

Similarly, we find:

$$F_\alpha(x+\epsilon) - F_\alpha(x-\epsilon) = 2\epsilon f_\alpha(x) + \frac{1}{2!}\int_x^{x+\epsilon} f_\alpha^{(2)}(t)(x+\epsilon-t)^2 dt + \frac{1}{2!}\int_{x-\epsilon}^{x} f_\alpha^{(2)}(t)(x-\epsilon-t)^2 dt$$

and so, by the same approach as above:

$$\left|\frac{1}{2!}\int_x^{x+\epsilon} f_\alpha^{(2)}(t)(x+\epsilon-t)^2 dt + \frac{1}{2!}\int_{x-\epsilon}^{x} f_\alpha^{(2)}(t)(x-\epsilon-t)^2 dt\right| \leq B\epsilon^3. \tag{2.1.9}$$

Combining (2.1.7) with (2.1.8) and (2.1.9) gives us:

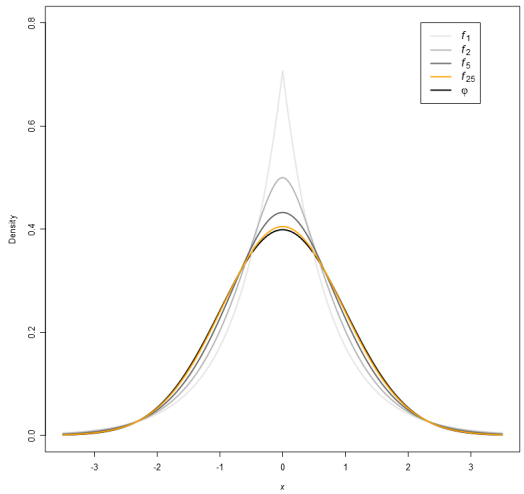


Figure 2.2: Comparison of f_α to ϕ.

$$
\begin{aligned}
2\epsilon^2 \geq & \left|\left(F_\alpha(x+\epsilon)-F_\alpha(x-\epsilon)\right)-\left(\Phi(x+\epsilon)-\Phi(x-\epsilon)\right)\right| \\
= & \Bigg| \left(2\epsilon f_\alpha(x)-2\epsilon\phi(x)\right) \\
& + \left(\left(\frac{1}{2!}\int_x^{x+\epsilon} f_\alpha^{(2)}(t)(x+\epsilon-t)^2dt + \frac{1}{2!}\int_{x-\epsilon}^{x} f_\alpha^{(2)}(t)(x-\epsilon-t)^2dt \right) \right. \\
& \left. - \left(\frac{1}{2!}\int_x^{x+\epsilon} \phi^{(2)}(t)(x+\epsilon-t)^2dt + \frac{1}{2!}\int_{x-\epsilon}^{x} \phi^{(2)}(t)(x-\epsilon-t)^2dt \right) \right) \Bigg|
\end{aligned}
$$

from which

$$
\begin{aligned}
& |2\epsilon f_\alpha(x)-2\epsilon\phi(x)| - B\epsilon^3 - B\epsilon^3 \\
\leq & \left|\left(F_\alpha(x+\epsilon)-F_\alpha(x-\epsilon)\right)-\left(\Phi(x+\epsilon)-\Phi(x-\epsilon)\right)\right| \\
\leq & 2\epsilon^2
\end{aligned}
$$

and hence

$$
|f_\alpha(x)-\phi(x)| \leq (1+B\epsilon)\epsilon \quad \forall\ \alpha > \alpha_\epsilon \tag{2.1.10}
$$

follows. Therefore, the CLT means that we can make f_α (which was shown to be equal to the kernel function of interest) arbitrarily close to ϕ by choosing α large enough. This then confirms (2.1.5) and is supported by Figure 2.2.

Computational Improvement

The next task is to make due on the claim of (2.1.4) offering a computational improvement over the quadratic cost associated with typical kernels (such as those in Figure 2.1). Plugging (2.1.4) into (2.1.3) (that which we seek to accelerate), we find:

$$
\sum_{i=1}^{n} K\left(\frac{x-x_i}{h}\right) w_i = \sum_{i=1}^{n} C_\alpha e^{-\frac{|x-x_i|}{h} d_\alpha} \sum_{k=0}^{\alpha-1} \beta_{\alpha k} \frac{|x-x_i|^k}{h^k} w_i
$$

$$
\begin{aligned}
&= \sum_{i=1}^{n} C_\alpha e^{-\frac{|x-x_i|}{h} d_\alpha} \left(\beta_{\alpha 0} + \beta_{\alpha 1} \frac{|x-x_i|}{h} + \ldots + \beta_{\alpha\alpha-1} \frac{|x-x_i|^{\alpha-1}}{h^{\alpha-1}} \right) w_i \\
&= C_\alpha \beta_{\alpha 0} \sum_{i=1}^{n} e^{-\frac{|x-x_i|}{h} d_\alpha} w_i + \frac{C_\alpha \beta_{\alpha 1}}{h} \sum_{i=1}^{n} |x-x_i| e^{-\frac{|x-x_i|}{h} d_\alpha} w_i + \ldots \\
&\quad + \frac{C_\alpha \beta_{\alpha\alpha-1}}{h^{\alpha-1}} \sum_{i=1}^{n} |x-x_i|^{\alpha-1} e^{-\frac{|x-x_i|}{h} d_\alpha} w_i. \qquad (2.1.11)
\end{aligned}
$$

As each of these terms require the same number of evaluations, it is sufficient to look at:

$$
\sum_{i=1}^{n} |x-x_i|^{\alpha-1} e^{-\frac{|x-x_i|}{h} d_\alpha} w_i, \qquad (2.1.12)
$$

since the computational cost of (2.1.11) is a constant multiple of that of (2.1.12).

Define $\{x_{i,n} : i = 1, 2, \ldots, n\}$ to represent the ordered x_i values (i.e. $x_{1,n} \leq x_{2,n} \leq \ldots \leq x_{n,n}$), and let $\{w_{i,n} : i = 1, 2, \ldots, n\}$ be the set of weight terms ordered according to the x_i values. Then, for each of $k = 0, 1, \ldots, \alpha - 1$ and $j = 0, 1, \ldots, n$, define:

$$
\mathrm{q}(k, j) = \sum_{i=1}^{j} (-x_{i,n})^k e^{\frac{x_{i,n}-x_{j,n}}{h} d_\alpha} w_{i,n}
$$

$$
\mathrm{r}(k, j) = \sum_{i=j+1}^{n} x_{i,n}^k e^{\frac{x_{j,n}-x_{i,n}}{h} d_\alpha} w_{i,n}
$$

with $\mathrm{q}(k, 0) = 0 = \mathrm{r}(k, n)$. Also, let $\mathrm{n}(x) = \sum_{i=1}^{n} \mathbb{1}(x_i \leq x)$. Then, consider the following reformulation from Hofmeyr (2019a):

$$
\begin{aligned}
&\sum_{i=1}^{n} |x-x_i|^{\alpha-1} e^{-\frac{|x-x_i|}{h} d_\alpha} w_i \\
&= \sum_{i=1}^{n} |x-x_{i,n}|^{\alpha-1} e^{-\frac{|x-x_{i,n}|}{h} d_\alpha} w_{i,n} \\
&= \exp\left(\frac{x_{\mathrm{n}(x),n} - x}{h} d_\alpha \right) \sum_{i=1}^{\mathrm{n}(x)} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} x^{\alpha-1-k} (-x_{i,n})^k e^{\frac{x_{i,n}-x_{\mathrm{n}(x),n}}{h} d_\alpha} w_{i,n} \\
&\quad + \exp\left(\frac{x - x_{\mathrm{n}(x),n}}{h} d_\alpha \right) \sum_{i=\mathrm{n}(x)+1}^{n} \sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} (-x)^{\alpha-1-k} x_{i,n}^k e^{\frac{x_{\mathrm{n}(x),n}-x_{i,n}}{h} d_\alpha} w_{i,n}.
\end{aligned}
$$

Swapping the order of summation and evaluating $x = x_{j,n}$, we find that (2.1.12) can be written in the form:

$$
\sum_{k=0}^{\alpha-1} \binom{\alpha-1}{k} \left(x_{j,n}^{\alpha-1-k} \, \mathrm{q}(k, j) + (-x_{j,n})^{\alpha-1-k} \, \mathrm{r}(k, j) \right), \qquad (2.1.13)
$$

which follows from $\mathrm{n}(x_{j,n}) = j$, giving $x_{\mathrm{n}(x_{j,n}),n} = x_{j,n}$. Finally, we note the recursive relationships:

$$
\mathrm{q}(k, j+1) = e^{\frac{x_{j,n}-x_{j+1,n}}{h} d_\alpha} \mathrm{q}(k, j) + (-x_{j+1,n})^k w_{j,n} \qquad (2.1.14)
$$

and

$$
\mathrm{r}(k, j-1) = e^{\frac{x_{j-1,n}-x_{j,n}}{h} d_\alpha} (\mathrm{r}(k, j) + x_{j,n}^k w_{i,n}). \qquad (2.1.15)
$$

The combination of (2.1.13) with (2.1.14) and (2.1.15) means that to evaluate (2.1.12) at each

of the order statistics requires computing $\mathrm{q}(k,j)$ and $\mathrm{r}(k,j)$ for some pair (k,j) – an operation needing n evaluations. Then, for any k, since we can recursively get the value at each j, we only need to iterate over $k = 0, 1, \ldots, \alpha - 1$. This recycling of terms brings the total number of evaluations to compute (2.1.12) to αn. Therefore, ignoring the cost of sorting the x_i terms, evaluating (2.1.3) at all of the sample points only costs $\frac{n\alpha(\alpha+1)}{2}$.

Combining the Above

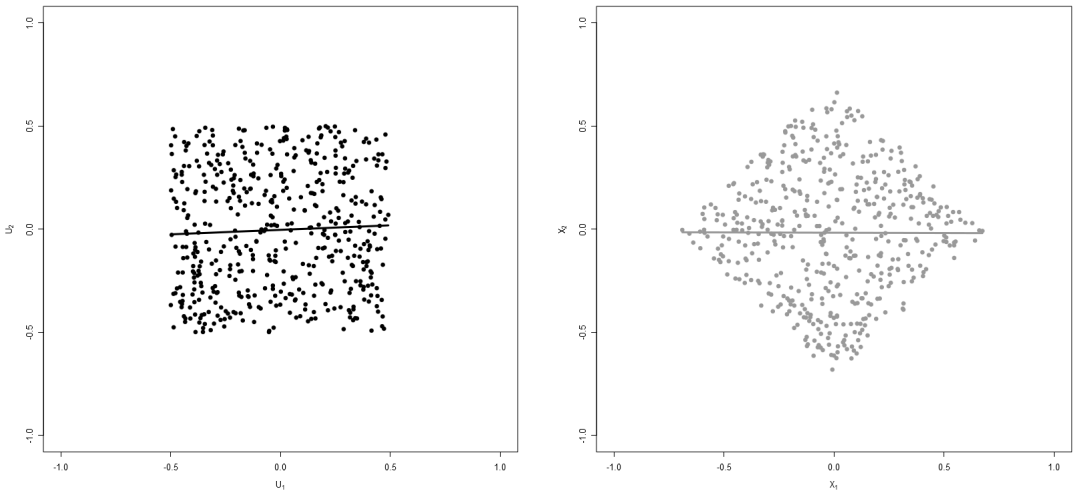
Presuming access to efficient sorting algorithms, and noting that the super-Gaussian nature associated with (2.1.4) (as highlighted in Figure 2.2) can be compensated for via making an appropriate bandwidth adjustment (implying that we can keep α small), we can conclude that the choice of kernel highlighted here can perform KDE that is very similar to taking $K = \phi$ in (2.1.3) while costing substantially less. This will be important in the optimisation context in which KDE is to be employed. It is noted that the $\alpha = 2$ is used in the sequel.

2.2 Independent Component Analysis

There are various situations in which a researcher is confined to indirect observation of some phenomena of interest. As an example, consider the field of sleep research. Described by Walker (2017:47-48), studying regions of the brain that are active during the different stages of sleep can be done via the placing of brainwave measuring electrodes on the scalp. Alas, this experiment is complicated by the fact that since an electrode cannot be made to target the output of a specific region, what is in actual fact measured is a convolution of independent underlying brainwaves. Of great interest, then, would be a procedure to deconvolve the signals – allowing researchers to study the summed contributions individually (although, this still leaves the problem of deciding which brain regions were responsible for each of the purified signals, accounting for the use of magnetic resonance imaging scans that do not suffer the same spatial drawbacks as do electrode measures). In our NB context, such a procedure would offer a way of manipulating available data into (better) meeting the model's assumptions, and hence improving its performance. This follows from the not unreasonable expectation that the purified signals, being assumed to have been generated by unrelated processes, should not share much statistical dependence.

Independent component analysis is concerned with exactly this problem. In its simple form, it assumes that we observe linear mixtures of latent sources called the "independent components", and its task is to extract these components from the available observations. The common thread among ICA methods is the objective of "non-Gaussianity" (discussed below) which can be achieved via a number of ways, including kurtosis, likelihood and information theoretic based procedures (Bressan and Vitrià, 2002:3). In this thesis, we shall focus on the latter of these methods. We note that the main source this section relies on is the work by Hyvärinen, Karhunen and Oja (2001).

Before commencing with this topic, we make a straightforward, yet crucial observation that continues with the discussion of independence introduced in Section 1.4. This observation concerns the relationship between independence and uncorrelatedness. Specifically, how there is only a one-way implication here: the former implies the latter, however (except in the special Gaussian case) the latter does not imply the former. Statistical independence is a much stronger condition than uncorrelatedness. In particular, uncorrelatedness can easily be achieved by methods such as principal component analysis (PCA), while ICA is a more challenging endeavor. To support this point, consider the plots in Figure 2.3. Here, we have simulated data from the joint distribution of two independent Uniform random variables (seen in the left panel of this figure). In the second window, the joint distribution of two linear mixtures of these Uniform variables, obtained via an orthogonal rotation, is given. Quite clearly, both figures illustrate uncorrelated variables (notice the added linear regression lines). However, in the second window, knowing the value of one of the variables does improve our ability to predict the other, and so these variables cannot be statistically independent. We will begin this section with a discussion on the information theoretic quantities of entropy and



(a) Independent Uniform variables. **(b)** Uncorrelated Uniform mixture.

Figure 2.3: Comparison of independence and uncorrelatedness.

mutual information. How these measures relate to Gaussianity and statistical independence is then discussed, followed by a brief presentation of the basics of ICA. Finally, the version of ICA relevant to NB (as introduced by Bressan and Vitrià (2002)) is discussed.

2.2.1 Entropy

The (differential) entropy, H, of a random variable is defined as:

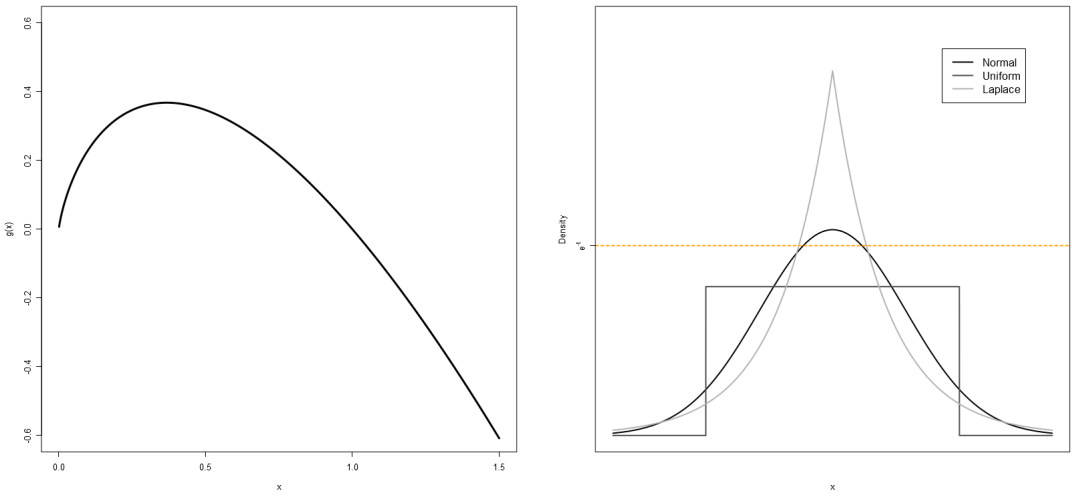
$$\begin{aligned} H(\mathrm{X}) &= -\,\mathrm{E}\left(\log\left(f_{\mathrm{X}}(\mathrm{X})\right)\right) \\ &= -\int_{-\infty}^{\infty} \log\left(f_{\mathrm{X}}(x)\right) f_{\mathrm{X}}(x)dx \\ &= \int_{-\infty}^{\infty} g\left(f_{\mathrm{X}}(x)\right) dx, \end{aligned} \tag{2.2.1}$$

where f_{X} is the probability density function of X, and $g(x) = -x\log(x)$. This quantity is said to be a measure of a variable's "randomness", or its (lack of) structure or predictability (Hyvärinen *et al.*, 2001:106). The easiest way to appreciate (2.2.1) as a measure of uncertainty, is to consider:

$$\mathrm{X} \sim \mathrm{Uniform}(-t, t).$$

Then, $H(\mathrm{X}) = \log(2t)$. Clearly, H is an increasing function of t: the parameter controlling the randomness of X.

As H is a measure of uncertainty, and as random variables that have larger variances can be considered "more uncertain", the function g (illustrated in the left panel of Figure 2.4) offers insight into what determines the size of (2.2.1) when keeping variance fixed. It can be shown that taking $x = e^{-1}$ maximises g, and it is clear that g is small when its argument is far away from this maximiser (and is also only positive for arguments between 0 and 1). Therefore, among random variables of equal variance, those having a PDF with values further from e^{-1} (either being much larger than e^{-1}, or close to 0) will have smaller entropy. This turns out to be the case for random variables that are heavily concentrated around certain values.



(a) Plot of g. **(b)** Equal variance densities.

Figure 2.4: Function related to entropy.

Before continuing, we note that the definition in (2.2.1) is naturally extended to the multi-dimensional case:

$$H(\mathbf{X}) = -\,\mathrm{E}\left(\log\left(f_{\mathbf{X}}(\mathbf{X})\right)\right), \tag{2.2.2}$$

where $f_{\mathbf{X}}$ is the PDF of the random vector, $\mathbf{X}$.

Now, a well-known result, one that is fundamental to what follows, is stated: among all distributions with a given covariance matrix, the Gaussian has maximum entropy (Hyvärinen *et al.*, 2001:112). This means that, accounting for variance, we can think of entropy not only as a measure of randomness, but also as one of similarity with Gaussianity: smaller entropy corresponding to "less Gaussianity".

In the right panel of Figure 2.4, we have presented the densities of three distributions of equal (unit) variance. As per the result given above, the Gaussian distribution (the black curve) is the one having maximum entropy of $\log\sqrt{2\pi}+1$ (the Uniform and Laplace variables have entropies of $\log\sqrt{2}$ and $\log\sqrt{2}+1$, respectively). What is interesting, is the fact that, although visually very different in the nature of their deviation from Gaussianity, the smaller entropy of the both the Uniform and Laplace random variables can be explained via the function g (introduced with (2.2.1)): the "spikiness" of the Laplace density, and the fact that the Uniform density is exactly zero for most of its domain, means that both random variables are contained to a relatively small interval with high probability (which is not the case for the Gaussian random variable). These are exactly the conditions for which g, and hence H, is small.

2.2.2 Mutual Information

Suppose that we have two random vectors, say $\mathbf{X}_1 \sim F_1$ and $\mathbf{X}_2 \sim F_2$, and that we were interested in the relatedness of their distributions. The Kullback-Leibler divergence, δ, is a measure that does this by assessing the (quasi-) distance between their density functions:

$$\delta\left(f_1, f_2\right) = \int_{\mathbb{R}^p} f_1(\boldsymbol{x})\,\log\left(\frac{f_1(\boldsymbol{x})}{f_2(\boldsymbol{x})}\right) d\boldsymbol{x}. \tag{2.2.3}$$

It can be shown that (2.2.3) is nonnegative, and that it is exactly equal to zero only in the case where $f_1 = f_2$. As can be seen from the definition, since (in general) $\delta\left(f_1, f_2\right) \neq \delta\left(f_2, f_1\right)$, the

Kullback-Leibler divergence is not a proper distance metric as it fails the property of symmetry. However, the closer in distribution $\mathbf{X}_1$ and $\mathbf{X}_2$, the smaller (2.2.3) will be.

The case of interest to us, and of relevance to this section, is when assessing (2.2.3) where $f_1 = f_{\mathbf{X}}$ and $f_2 = \prod_{j=1}^{p} f_{\mathrm{X}_j}$. That is, the comparison of the joint density of the random vector,

$$\mathbf{X} = \begin{bmatrix} \mathrm{X}_1 \\ \mathrm{X}_2 \\ \vdots \\ \mathrm{X}_p \end{bmatrix},$$

to that of the product of the marginal densities of its components. Using these choices of f_1 and f_2, from the definition (2.2.3), we find:

$$\begin{aligned}
\delta(f_1, f_2) &= -\int_{\mathbb{R}^p} f_1(\boldsymbol{x}) \log(f_2(\boldsymbol{x}))\, d\boldsymbol{x} + \int_{\mathbb{R}^p} f_1(\boldsymbol{x}) \log(f_1(\boldsymbol{x}))\, d\boldsymbol{x} \\
&= -\int_{\mathbb{R}^p} f_{\mathbf{X}}(\boldsymbol{x}) \log\left(\prod_{j=1}^{p} f_{\mathrm{X}_j}(x_j)\right) d\boldsymbol{x} + \int_{\mathbb{R}^p} f_{\mathbf{X}}(\boldsymbol{x}) \log(f_{\mathbf{X}}(\boldsymbol{x}))\, d\boldsymbol{x} \\
&= -\int_{\mathbb{R}^p} f_{\mathbf{X}}(\boldsymbol{x}) \left(\sum_{j=1}^{p} \log\left(f_{\mathrm{X}_j}(x_j)\right)\right) d\boldsymbol{x} + \mathrm{E}(\log(f_{\mathbf{X}}(\mathbf{X}))) \\
&= \sum_{j=1}^{p} \left(-\int_{\mathbb{R}^p} f_{\mathbf{X}}(\boldsymbol{x}) \log\left(f_{\mathrm{X}_j}(x_j)\right) d\boldsymbol{x}\right) - H(\mathbf{X}) \\
&= \sum_{j=1}^{p} \left(-\int_{-\infty}^{\infty} f_{\mathrm{X}_j}(x_j) \log\left(f_{\mathrm{X}_j}(x_j)\right) dx_j\right) - H(\mathbf{X}) \\
&= \sum_{j=1}^{p} H(\mathrm{X}_j) - H(\mathbf{X}) \\
&=: I(\mathbf{X}).
\end{aligned} \tag{2.2.4}$$

In this special case, (2.2.3) is equal to the so-called mutual information, I, of $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_p$: a measure that compares the information available from the marginal densities to that present in the full joint distribution (importantly, mutual information does this by taking into account the whole dependence structure of the variables – not only their covariances). Keeping in mind that the Kullback-Leibler divergence is a (quasi-) distance, we can consider (2.2.4) as a measure of the dependence of $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_p$, and we see that $I(\mathbf{X}) = 0$ if only if $f_{\mathbf{X}} = \prod_{j=1}^{p} f_{\mathrm{X}_j}$, i.e. when these variables are statistically mutually independent. Furthermore, the closer these variables are to independence, the smaller we can expect (2.2.4) to be.

2.2.3 The Independent Component Analysis Model

ICA is typically captured via a "latent" variable model. The assumption made is that we have access to linear mixtures of underlying sources of interest. That is, we can observe realisations of the random variables $\mathrm{X}_1, \mathrm{X}_2, \ldots, \mathrm{X}_p$, where it is assumed that:

$$\mathrm{X}_j = \sum_{i=1}^{p} a_{ji} \mathrm{S}_i = \mathbf{a}_j^{\top} \mathbf{S} \qquad j = 1, 2, \ldots, p, \tag{2.2.5}$$

with $a_{ji} \in \mathbb{R}$ being the unknown mixing values. Access to the sources, $\{\mathrm{S}_j : j = 1, 2, \ldots, p\}$, which are assumed to be independent, is indirect via the X_j, and recovering them is the focus of ICA. The classic example of ICA is the so-called "cocktail party" problem: isolating the speech of partygoers recorded from a collection of microphones scattered over the venue.

Prominent areas of its application include econometrics, brain imaging and feature extraction. With Naïve Bayes in mind, the latter of these is of interest to us.

ICA works by making very few assumptions. No information on the distributions of the unknown sources are required. The only necessary assumptions are that of statistical independence and that the sources is non-Gaussian. Writing (2.2.5) in matrix form, we find:

$$\mathbf{X} = \boldsymbol{A}\mathbf{S},$$

where

$$\boldsymbol{A} = \begin{bmatrix} \mathbf{a}_1^\top \\ \mathbf{a}_2^\top \\ \vdots \\ \mathbf{a}_p^\top \end{bmatrix}.$$

The task is to learn the mixing matrix, $\boldsymbol{A}$, and then to recover the independent sources as:

$$\mathbf{S} = \boldsymbol{B}\mathbf{X}, \tag{2.2.6}$$

where $\boldsymbol{B} = \boldsymbol{A}^{-1}$ (with it being necessary to assume invertibility for $\boldsymbol{A}$). The learning of the unknown matrix can be done in a number of ways. As mentioned in the introduction to this section, approaches include kurtosis, likelihood and information theoretic based procedures.

2.2.4 Independent Component Analysis and Naïve Bayes

For our purposes, we will alter the above model slightly: the matrix $\boldsymbol{B}$ will be learnt directly. We also won't assume that the sources are strictly independent, but rather focus on finding $\{b_{ij} \in \mathbb{R} : j = 1, 2, \ldots, p\}$ such that the variables

$$\mathrm{S}_j = \sum_{i=1}^{p} b_{ji}\mathrm{X}_i = \boldsymbol{b}_j^\top \mathbf{X} \qquad j = 1, 2, \ldots, p' \leq p,$$

are as independent as possible. If we assume for the moment that $\boldsymbol{B}$ is square (i.e., $p = p'$), then we can consider the mutual information of the components of $\mathbf{S}$. Having established (2.2.4) as a measure of independence, it follows:

$$\begin{aligned} I(\mathbf{S}) &= \sum_{j=1}^{p'} H(\mathrm{S}_j) - H(\mathbf{S}) \\ &= \sum_{j=1}^{p'} H(\mathrm{S}_j) - H(\mathbf{X}) - \log|\det \boldsymbol{B}|, \end{aligned} \tag{2.2.7}$$

where the result for the entropy of a linear transformation of a random vector has been used (Hyvärinen *et al.*, 2001:109).

With the goal of maximising independence of the components of $\mathbf{S}$, our objective is then to find $\boldsymbol{B}$ such that (2.2.7) is minimised. Under the constraint $\boldsymbol{B}^\top\boldsymbol{B} = \boldsymbol{I}_p$, it must be that $\det \boldsymbol{B}^\top\boldsymbol{B} = (\det \boldsymbol{B})^2 = \det \boldsymbol{I}_p = 1$, and hence that $\det \boldsymbol{B}$ is constant. Furthermore, as $H(\mathbf{X})$ is also constant with respect to $\boldsymbol{B}$, we have:

$$I(\mathbf{S}) = \sum_{j=1}^{p'} H(\mathrm{S}_j) + c, \tag{2.2.8}$$

where $c \in \mathbb{R}$ is some constant not dependent on $\boldsymbol{B}$. From (2.2.8) we see that finding the entries in $\boldsymbol{B}$ that maximises the independence of the components of $\mathbf{S}$ is equivalent to minimising the sum of entropies of the S_i variables (under the constraint $\boldsymbol{B}^\top\boldsymbol{B} = \boldsymbol{I}_p$). Recalling the discussion in Section 2.2.1 on the connection between (2.2.1) and Gaussianity, we see that there is a close

relationship between maximising the statistical mutual independence of $\{S_j : j = 1, 2, \ldots, p'\}$ and minimising a measure of the total Gaussianity. The pursuit of independence can thus be interpreted as an objective of maximising "non-Gaussianity".

The quantity (2.2.8) can also be considered for the case where $p' < p$. With feature extraction in mind, and in a statistical learning context, this would imply a dimension reduction which could potentially have a favourable impact on accuracy. Although it also means violation of the steps from (2.2.7) to (2.2.8), we would still expect minimising the sum of mutual information to move $\{S_j : j = 1, 2, \ldots, p'\}$ away from Gaussianity and toward independence. Doing so in this manner remains well motivated.

The application of ICA to NB is the contribution of Bressan and Vitrià (2002). This worked from the reasonable expectation that the improved validity of the assumption underlying NB implies better classification accuracy. That is, looking back to the framework introduced in Section 1.2.1, all other things being equal, we expect NB to perform better on data-sets where the independence of the conditional random variables, $\{X_j \mid Y = k : j = 1, 2, \ldots, p\}$ for each of $k = 1, 2, \ldots, K$ is more reasonable. In fact, Palacios-Alonso, Brizuela and Sucar (2010) made this claim: in domains where the independence assumption of NB is not met, classification accuracy deteriorates. Bressan and Vitrià's proposal takes advantage of exactly this observation, and sought to artificially create the favourable circumstances of class conditional independence. This artificial recreation was achieved by coercing the input features into matching the NB assumption via a CC ICA representation. Their idea was to extract feature sets $\{S_j^k : j = 1, 2, \ldots, p'\}$ for each of $k = 1, 2, \ldots, K$. This involved learning K separate unmixing matrices, $\boldsymbol{B}^k$, via the methodology introduced in this section. The exact workings of doing so (specifically, estimation and optimisation of (2.2.8)) is covered in Section 3.4. Section 4.1.1 describes the practical details of using CC ICA together with NB. In Section 4.2 and 4.3 we benchmark the performance of CC ICA against that of the method proposed in this thesis. At this point it is merely noted that Bressan and Vitrià reported a significant improvement of the CC ICA formulation over both the original and PCA representation on standard data-sets.

2.3 Optimisation

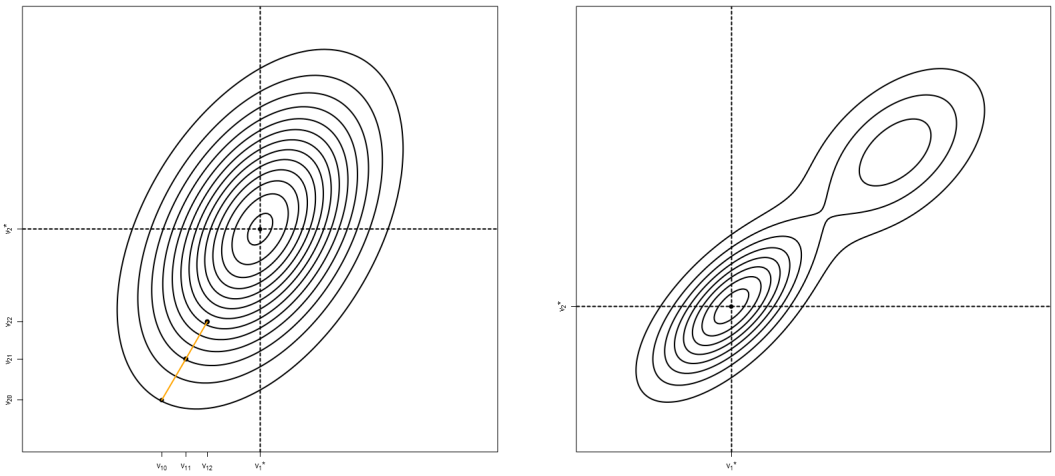
Numerical methods are necessary to deal with optimisation tasks that cannot be solved analytically. E.g., faced with finding the unmixing matrix, $\boldsymbol{B}$, when performing an independent component analysis presents a difficulty – there is no closed form solution for the matrix minimising (2.2.8) (ignoring the constraint of $\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}_p$ for the moment). Problems of this kind are no small matter and there is a vast collection of literature dedicated to the topic (see Boyd and Vandenberghe (2013) for a thorough treatment). This section briefly looks at one such numerical optimisation technique, and then pays attention to a variant of this technique that was proposed, independently, by each of Broyden, Fletcher, Goldfarb and Shanno in 1970 (Papakonstantinou (2009:65-72) provides a full historical account of this development).

2.3.1 Gradient Ascent

A classic numerical optimisation procedure is gradient (or steepest) ascent. Suppose we have an objective function of interest, say $C : \mathbb{R}^2 \mapsto \mathbb{R}$, and we wish to find the argument, $\boldsymbol{v}^*$, such that $C(\boldsymbol{v}) \leq C(\boldsymbol{v}^*)$ for all $\boldsymbol{v} \in \mathbb{R}^2$. Steepest ascent works by starting-off with some initial estimate (or guess) of $\boldsymbol{v}^*$, say $\boldsymbol{v}_0$, and then repeatedly updating this estimate in the direction of the gradient of the objective function, C, in search of $\boldsymbol{v}^*$:

$$\boldsymbol{v}_t = \boldsymbol{v}_{t-1} + \eta_t \frac{\partial C(\boldsymbol{v})}{\partial \boldsymbol{v}}\Big|_{v=v_{t-1}} \qquad t = 1, 2, \ldots, \tag{2.3.1}$$

with the learning rate (or, step-size), η_t, determining the size of the step taken in the gradient direction (note that this is a dynamic quantity that can be adjusted as the iterative process continues). The procedure in (2.3.1) is terminated when some stopping criterion is satisfied (e.g., when $\|\boldsymbol{v}_t - \boldsymbol{v}_{t-1}\| < \epsilon$ where $\epsilon > 0$ is some small amount).



(a) Steepest ascent. **(b)** Non-concave objective.

Figure 2.5: Optimisation illustration.

A simple way to appreciate the utility of relying on the gradient in this way is to consider the optimisation of the objective depicted by the contours in the left panel of Figure 3.1. Here, the starting value, v_0, as well as the first two steps of gradient ascent, v_1 and v_2, have been indicated. Graphically, we see that relying on the gradient means taking steps that are perpendicular to the contours. From this, we see that moving in the direction of the gradient is sensible.

The success of gradient ascent will be dependent on the concavity of the objective. Mathematically, a function, say $g : \mathbb{R}^p \mapsto \mathbb{R}$, is defined to be "concave" if it holds that:

$$g\left((1-\theta)\boldsymbol{x}+\theta\boldsymbol{y}\right) \geq (1-\theta)\, g\left(\boldsymbol{x}\right)+\theta g\left(\boldsymbol{y}\right) \qquad \forall\ \boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^p$$

for any $\theta \in [0,1]$. With a non-concave objective function, there is a risk of getting stuck at some local optimum (see the right panel of Figure 3.1). This is a serious problem. It is very likely that we will face functions that have many local- maxima and/or minima. Choosing a good starting value (i.e., the choice of v_0) goes a long way in addressing this issue. Another crucial setting regarding the convergence (and the rate thereof) of gradient ascent is the learning rate. Inspecting Figure 2.5, one can imagine how a poor choice of η_t (particularly choosing η_t to be too large) can induce a situation where the estimate of v^* oscillates around inferior values.

2.3.2 The Broyden, Fletcher, Goldfarb and Shanno Algorithm

The difficulty of the optimal choice of step-size i.e., the optimal choice of sequence $\{\eta_t : t = 1,2,\ldots\}$ and starting value, v_0, motivates some extension to gradient ascent. Second-order optimisation methods amend (2.3.1) by supplementing the update process with the second derivative of the objective function. The matrix of partial derivatives, $\frac{\partial^2 C(v)}{\partial v^2}$, captures information on the curvature of C. This can be used to inform the direction in which a step should be taken (Hyvärinen *et al.*, 2001:65). Newton's method works by swapping (2.3.1) for the update:

$$v_t = v_{t-1} + \left(\frac{\partial^2 C\left(v\right)}{\partial v^2}\Big|_{v=v_{t-1}}\right)^{-1} \frac{\partial C\left(v\right)}{\partial v}\Big|_{v=v_{t-1}} \qquad t = 1,2,\ldots\,.$$

Under suitable conditions, this will deliver fast convergence (see Kohonen (1996) for a full discussion of Newton's method and its convergence properties). However, computing the

second derivative and its inverse can be a computationally expensive exercise, making this algorithm infeasible in many applications.

BFGS is a popular quasi-Newton technique (Nawi, Ransing & Ransing, 2006:154). It is a procedure that efficiently approximates the inverse of the second derivative of the objective function (i.e., the Hessian), and so mitigates the computational problem of Newton's method (Byatt, Coope & Price, 2004). This is the numerical optimisation procedure of choice in the sequel.

2.4 Conclusion

This chapter presented the technique of kernel density estimation. We motivated its use, and looked at an extension that addresses the computational burden of the standard approach. Independent component analysis was offered as a response to the problem of only having indirect access to a collection of components of interest. To this end, we looked into the information theoretic concepts of entropy, the Kullback-Leibler divergence and the mutual information of a collection of random variables. ICA was presented in the context of the naïve Bayes classifier. Specifically, it was seen how a class conditional application of ICA can be used to improve the validity of the NB assumption of CC independence. The final section briefly discussed the algorithm due to Broyden, Fletcher, Goldfarb and Shanno as a computationally efficient numerical optimisation procedure.

Chapter 3

Projected Naïve Bayes

This chapter details our proposal for improving the naïve Bayes classifier. We seek to learn a projection matrix that delivers a feature representation improving classification accuracy. The maximum likelihood formulation of this problem is presented. This is followed by a discussion of the implementation and optimisation practicalities of the method. The improvement is first discussed under the common NB assumption of Gaussianity. That is, the learnt set of features are initially modeled as class conditionally having Gaussian distributions, giving the suitably called Projected Gaussian NB (PGNB) classifier. Subsequently, the (strong) parametric assumption is relaxed. The Gaussian treatment is swapped for kernel density estimation, giving the Projected Kernel NB (PKNB) classifier. The penultimate section of the chapter establishes a link between our proposal and independent component analysis. The projection matrix is argued to have an independence inducing effect similar to that of the method due to Bressan and Vitrià (2002). Finally, Projected NB is discussed in light of the bias-variance trade-off. Attention is also paid to the dimension reduction and visualisation potential of the method.

This chapter will rely on the notation and terminology introduced in Chapter 1. The reader will be regularly referred back to Sections 1.2.1 and 1.2.2. We develop the proposal under the assumption that the input features (i.e., the conditional random variables, $X_j \mid Y = k$) are continuous. It has already been established that the pursuit of improving naïve Bayes has a long history, is well-motivated and widely accepted, and is one that is worthy of our efforts. This chapter adds to those existing attempts highlighted in Section 1.5.

3.1 The Learning of a Projection Matrix

Section 2.2 suggests that learning an alternative representation of data can be advantageous. With independent component analysis, this advantage is found in improved understanding of underlying components of interest (assumed to be only indirectly observable). With NB, an alternative representation of data could mean addressing violations of its assumption in the original representation. This is done under the belief that the closer a particular data-set matches the assumed class conditional mutual statistical independence (see the discussion around (1.2.2)), the better NB should be at accurately classifying input patterns. As was the case with ICA, the alternative representation we develop for NB is specified via linear combinations of the original inputs. This makes the task of learning said representation much easier than would be the case under a non-linear transformation as it avoids dealing with the difficulty of specifying the non-linear transformation, as well as the associated optimisation implied by a non-linear transformation. In addition, in line with the discussion given in Section 3.5.2 at the end of this chapter, a non-linear transformation might imply too many degrees of freedom. This has the potential of trading too biased a classifier (i.e., the standard NB) for one that is too variable. Furthermore, unlike the class conditional ICA of Bressan and Vitrià (2002), we learn a single transformation rather than K separate unmixing matrices. This delivers a more parsimonious and interpretable classifier, and also has advantageous visualisation implications. That is, the learnt transformation allows visualisation of the data which exposes the discrimination of classes – something which cannot readily be achieved when having K distinct unmixing matrices.

3.1.1 The Likelihood Function

The idea of a likelihood function is to measure the plausibility of an observed outcome given a particular set of parameters. Recalling that the philosophy of NB is to allocate an observation, say $\boldsymbol{x}$, to the class maximising the posterior probability, $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$, and keeping in mind the assumed access to a set of n observed samples, $\{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$, the appropriate likelihood function is given as the conditional probability of the observed outcome:

$$L = \prod_{i=1}^{n} \mathrm{P}\left(\mathrm{Y} = y_i \mid \boldsymbol{x}_i\right), \tag{3.1.1}$$

where L can be considered a function of the parameters that characterise the probability terms, P. Looking at the above, it follows that improvement of the discriminatory ability of NB should come about when the constituent terms involved in (3.1.1) are increased in their magnitude (i.e., moved closer to 1). Maintaining our frequentist perspective, this follows since it means that (on the sample data, at least) the classifier has improved certainty of the class to which a particular observation belongs. Larger constituent terms will correspond to larger values of L, and so it is reasonable to want to find those parameters maximising (3.1.1): the ones that make the observed outcome "most likely" or "most probable" (Rice, 2007:267).

Section 1.3.1 discusses NB along the lines of the bias-variance trade-off. It was established that the success of NB can be explained according to the low variance implication of the assumption of CC mutual statistical independence. It was also settled that this low variance comes at the cost of potentially biased estimates of the relevant probability terms, but that the effect of slight bias might not be problematic since we only need (1.3.3) to hold. That is, the only condition necessary for success (Rish, 2001:1) is that both the estimated and actual probability terms agree on the class that is most likely. However, a structural bias can still be expected to have a negative impact on the overall performance of the classifier, and so we have reason to address it. The following, then, appears to be a reasonable inquiry: is it possible to manoeuvre the bias-variance trade-off to a position that is more favourable in a classification accuracy sense? This is the thinking behind the improvement that now follows.

3.1.2 Projected Naïve Bayes

The approach we propose is to learn a projection (or, transformation) of the data that improves classification accuracy under NB. This is done, following the discussion of Section 3.1.1 above, with the link between improving classification accuracy and increasing the likelihood in mind. To this end, consider the introduction of a projection matrix:

$$\boldsymbol{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1p'} \\ v_{21} & v_{22} & \cdots & v_{2p'} \\ \vdots & \vdots & \ddots & \vdots \\ v_{p1} & v_{p2} & \cdots & v_{pp'} \end{bmatrix} = \begin{bmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \cdots & \boldsymbol{v}_{p'} \end{bmatrix},$$

where $\boldsymbol{V} \in \mathbb{R}^{p \times p'}$ and $p' \leq p$ (where p is the total number of predictors), for the task of feature extraction. What we want is to learn this matrix such that it can be used for the replacement of the input pattern, $\boldsymbol{x}$, with the transformed (or, projected) counterpart, $\boldsymbol{V}^\top \boldsymbol{x}$. The hope in doing so is that the success rate of

$$\underset{k=1,2,\ldots,K}{\arg\max}\ \mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x}) = \underset{k=1,2,\ldots,K}{\arg\max}\ \hat{\mathrm{P}}\left(\mathrm{Y} = k \mid \boldsymbol{x}, \boldsymbol{V}\right) \tag{3.1.2}$$

is better than that which is achieved when the estimation of $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$ occurs according to Section 1.2.2 (i.e., better than the standard NB). The entries in $\boldsymbol{V}$ can be thought of as parameters. With the paradigm of Section 1.3.1 in mind, these parameters are hoped to provide the additional "freedom" needed to effect improved classification accuracy by making a favourable trade-off between bias and variance.

What is proposed is to swap the input patterns stored in $\boldsymbol{X}$ (as presented in (1.2.1)) for the alternative representation:

$$\underset{n\times p'}{\boldsymbol{XV}} = \begin{bmatrix} \left(\boldsymbol{V}^\top \boldsymbol{x}_1\right)^\top \\ \left(\boldsymbol{V}^\top \boldsymbol{x}_2\right)^\top \\ \vdots \\ \left(\boldsymbol{V}^\top \boldsymbol{x}_n\right)^\top \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1^\top \boldsymbol{v}_1 & \boldsymbol{x}_1^\top \boldsymbol{v}_2 & \dots & \boldsymbol{x}_1^\top \boldsymbol{v}_{p'} \\ \boldsymbol{x}_2^\top \boldsymbol{v}_1 & \boldsymbol{x}_2^\top \boldsymbol{v}_2 & \dots & \boldsymbol{x}_2^\top \boldsymbol{v}_{p'} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{x}_n^\top \boldsymbol{v}_1 & \boldsymbol{x}_n^\top \boldsymbol{v}_2 & \dots & \boldsymbol{x}_n^\top \boldsymbol{v}_{p'} \end{bmatrix},$$

where $\boldsymbol{x}_i^\top \boldsymbol{v}_j = \sum_{t=1}^{p} x_{it} v_{jt}$. This is reminiscent of the search for the unmixing matrix, $\boldsymbol{B}$, with the discussion of ICA. Rather than wanting to learn a representation of the data that delivers independence, our focus now is that of achieving (3.1.2), which is done according to the likelihood objective given in (3.1.3), and elaborated, below.

As was the case in Section 2.2, the search for the projection matrix needs to be conducted according to a well-motivated objective. Remarking that the joint distribution, $(\mathbf{X}, \mathrm{Y})$, is unknown, we cannot find the true likelihood of the transformed data. However, seeing as the purpose of the proposed method is to improve NB, we can formulate the likelihood under the assumptions of NB. This gives:

$$\begin{aligned} L(\boldsymbol{V}) &= \prod_{i=1}^{n} \mathrm{P}\left(\mathrm{Y} = y_i \mid \boldsymbol{x}_i, \boldsymbol{V}\right) \\ &= \prod_{i=1}^{n} \frac{\left(\prod_{t=1}^{p'} f_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{v}_t\right)\right) \pi^{y_i}}{\sum_{k=1}^{K} \left(\prod_{t=1}^{p'} f_t^{k}\left(\boldsymbol{x}_i^\top \boldsymbol{v}_t\right)\right) \pi^{k}}, \end{aligned} \tag{3.1.3}$$

where the fact that L depends on $\boldsymbol{V}$ has been indicated explicitly, and with the law of total probability relied on for the expansion of the denominator term. It follows from the discussion above that increasing (3.1.3) implies an improved fit of our model to the sample data.

When learning a projection matrix of this kind it is standard to enforce some constraint on the magnitude of its entries. Defining $\|.\|$ as the ℓ_2-norm of a real-valued vector, let $\boldsymbol{u}_t = \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$. Then, from the fact that $\log(\cdot)$ is strictly monotonically increasing, we trade (3.1.3) for the mathematically convenient:

$$\begin{aligned} \ell(\boldsymbol{U}) &= \log\left(L(\boldsymbol{U})\right) \\ &= \sum_{i=1}^{n} \log\left(\left(\prod_{t=1}^{p'} f_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right) \pi^{y_i}\right) - \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \left(\prod_{t=1}^{p'} f_t^{k}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right) \pi^{k}\right), \end{aligned} \tag{3.1.4}$$

with $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_{p'} \end{bmatrix}$ denoting the normalised version of the projection matrix. The solution delivered when maximising $\ell(\boldsymbol{U})$ with respect to $\boldsymbol{V}$ will differ from that maximising (3.1.3) only in that its columns are restricted to having unit norm. We note that another typical constraint when facing optimisation tasks of this kind is to enforce the columns of the projection matrix to be orthogonal. However, we seek to find a projection where the optimisation is driven by the supervision due to the labels, $\boldsymbol{y}$, as much as possible, and so will not enforce orthogonality (although Section 3.4 does argue that there is a link between orthogonality and the role of the second term in (3.1.4)).

The main concern of the proposal introduced here is how to go about learning the projection matrix, $\boldsymbol{V}$, that maximises (3.1.4). Since, however, the quantities π^k and f_t^k are unknown, we have to settle for finding a projection which maximises an estimate of (3.1.4). To do so will require a strategy regarding the unknown quantities above: π^k and f_t^k. As per the argument of consistency in Section 1.2.2, estimation of $\pi^k = \mathrm{P}(\mathrm{Y} = k)$ will be done according

to (1.2.3) (for which we have almost sure convergence to the actual value). The function f_t^k now denotes the probability density function of the t^{th} transformed predictor conditional on $\mathrm{Y} = k$ (i.e., the density function of the class conditional random variable: $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$). Both a parametric and nonparametric treatment of f_t^k will be considered in this dissertation. The former of the two now follows, with the latter postponed to Section 3.3.

3.2 Projected Gaussian Naïve Bayes

This section looks at the treatment of $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$ as a Gaussian stochastic variable. We estimate the objective in (3.1.4) in accordance with assuming said Gaussianity, and look at some of the details of applying the BFGS algorithm to the task of learning the projection matrix, $\boldsymbol{V}$. The classifier developed is called Projected Gaussian naïve Bayes.

3.2.1 Posterior Treatment

The marginal distributions of the conditional random vector, $\mathbf{X} \mid \mathrm{Y} = k$, are unknown for each of $k = 1, 2, \ldots, K$. Hence, the distribution of the transformed quantity, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$, is unknown. This shortcoming means that some estimation strategy for the unknown density function, f_t^k, is needed. Borrowing from the common naïve Bayes strategy that models a continuous random variable defined on $\mathbb{R}$ as a Gaussian (John & Langley, 1995:339), leads to the assumption:

$$\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k \sim \mathrm{N}\left(\mu_t^k, \sigma_{tt}^k\right) \qquad t = 1, 2, \ldots, p', \tag{3.2.1}$$

where $\mathrm{E}\left(\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y}\right) = \mu_t^k$ and $\mathrm{var}\left(\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y}\right) = \sigma_{tt}^k$ denotes the mean and variance of the transformed random variable. This parameterisation makes the problem of estimating f_t^k much more manageable. We now merely need to decide on how to approximate $\{\mu_t^k, \sigma_{tt}^k : t = 1, 2, \ldots, p'\}$ for each of the K classes. From the assumption of Gaussianity, and since we take $\{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$ to be the realisations of n independent and identically distributed random variables, $(\mathbf{X}, \mathrm{Y}) \sim F_{(\mathbf{X},\mathrm{Y})}$, a likelihood function can be constructed:

$$L\left(\mu_t^k, \sigma_{tt}^k\right) = \prod_{i:y_i=k} \frac{1}{\sqrt{2\pi\sigma_{tt}^k}} \exp\left(\frac{-\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t - \mu_t^k\right)}{2\sigma_{tt}^k}\right).$$

The parameters maximising $L\left(\mu_t^k, \sigma_{tt}^k\right)$ can easily be found (as it were with (1.2.7) and (1.2.8)), giving the sensible estimates:

$$\hat{\mu}_t^k = \frac{1}{n^k} \sum_{i:y_i=k} \boldsymbol{x}_i^\top \boldsymbol{u}_t \tag{3.2.2}$$

and

$$\hat{\sigma}_{tt}^k = \frac{1}{n^k} \sum_{i:y_i=k} \left(\boldsymbol{x}_i^\top \boldsymbol{u}_t - \hat{\mu}_t^k\right)^2, \tag{3.2.3}$$

where we recall $n^k = \sum_{i=1}^n \mathbb{1}(y_i = k)$ and that $\boldsymbol{u}_t = \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$.

Combining the parametric assumption on $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$ with the estimates (3.2.2) and (3.2.3), we state the objective function as:

$$\ell(\boldsymbol{U}) = \sum_{i=1}^n \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right) \hat{\pi}^{y_i}\right) - \sum_{i=1}^n \log\left(\sum_{k=1}^K \left(\prod_{t=1}^{p'} \hat{f}_t^k(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right) \hat{\pi}^k\right), \tag{3.2.4}$$

where the notation suppresses acknowledgment of the parameter estimates $\{\hat{\mu}_t^k, \hat{\sigma}_{tt}^k : t = 1, 2, \ldots, p'\}$ for $k = 1, 2, \ldots, K$. Looking at (3.2.4), and noting that $\boldsymbol{U} = \begin{bmatrix}\boldsymbol{u}_1 & \boldsymbol{u}_2 & \ldots & \boldsymbol{u}_{p'}\end{bmatrix}$, it

is not possible to directly solve for the matrix, say $\boldsymbol{V}^*$, where $\ell(\boldsymbol{U}) \leq \ell(\boldsymbol{U}^*)$, and so we must rely on some numerical strategy such as that outlined in Section 2.3 to optimise (3.2.4).

3.2.2 Optimisation: BFGS

Following Section 2.3.2, Newton's method in terms of our objective function, $\ell : \mathbb{R}^{p \times p'} \mapsto \mathbb{R}$, is given as:

$$\boldsymbol{V}_t = \boldsymbol{V}_{t-1} + \left(\frac{\partial^2 \ell(\boldsymbol{U})}{\partial \boldsymbol{V}^2} \Big|_{\boldsymbol{U}=\boldsymbol{U}_{t-1}} \right)^{-1} \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}} \Big|_{\boldsymbol{U}=\boldsymbol{U}_{t-1}} \qquad t = 1, 2, \ldots$$

where $\frac{\partial^2 \ell(\boldsymbol{U})}{\partial \boldsymbol{V}^2}$ is the array of second derivatives and noting again that the matrix $\boldsymbol{U}$ is equal to that of $\boldsymbol{V}$ with ℓ_2-normalised columns. Computing the Hessian and its inverse is a costly exercise, and doing this over many iterations becomes infeasible. Remark that, practically, we vectorise $\boldsymbol{V}$ in order for the Hessian to be stored in matrix form. It was noted that BFGS mitigates the computational issues of Newton's method by efficiently approximating the inverse of the Hessian. However, it still requires exact values of the first derivative, $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}}$, for each update step of the procedure. Hence, its efficient computation is of concern.

Computation of $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}}$

Evaluating $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}} = \begin{bmatrix} \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_1} & \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_2} & \cdots & \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_{p'}} \end{bmatrix}$, with each column here denoting a vector gradient, directly follows as:

$$\begin{aligned} \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_z} &= \sum_{i=1}^{n} \left(\frac{\partial}{\partial \boldsymbol{v}_z} \log \left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^{y_i} \right) - \frac{\partial}{\partial \boldsymbol{v}_z} \log \left(\sum_{k=1}^{K} \left(\prod_{t=1}^{p'} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^{k} \right) \right) \\ &= \sum_{i=1}^{n} \left(\frac{\frac{\partial}{\partial \boldsymbol{v}_z} \hat{f}_z^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_z)}{\hat{f}_z^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_z)} - \frac{\sum_{k=1}^{K} \left(\left(\prod_{t \neq z} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^k \frac{\partial}{\partial \boldsymbol{v}_z} \hat{f}_z^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_z) \right)}{\sum_{k=1}^{K} \left(\left(\prod_{t=1}^{p'} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^k \right)} \right). \end{aligned} \tag{3.2.5}$$

From (3.2.5) it is clear that computing $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}}$ comes down to finding $\frac{\partial}{\partial \boldsymbol{v}_z} \hat{f}_z^k$. To assist in doing so, for each of $k = 1, 2, \ldots, K$, define $\bar{\boldsymbol{x}}^k$ and $\boldsymbol{S}^k$ to be the sample mean and covariance quantities related to those values stored in $\boldsymbol{X}$ which belong to the k^{th} class (introduced in Section 1.2.1):

$$\bar{\boldsymbol{x}}^k = \begin{bmatrix} \bar{x}_1^k \\ \bar{x}_2^k \\ \vdots \\ \bar{x}_p^k \end{bmatrix}$$

and

$$\boldsymbol{S}^k = \begin{bmatrix} s_{11}^k & s_{12}^k & \cdots & s_{1p}^k \\ s_{21}^k & s_{22}^k & \cdots & s_{2p}^k \\ \vdots & \vdots & \ddots & \vdots \\ s_{p1}^k & s_{p2}^k & \cdots & s_{pp}^k \end{bmatrix},$$

where $\bar{x}_j^k = \frac{1}{n^k}\sum_{i:y_i=k} x_{ij}$ and $s_{tj}^k = \frac{1}{n^k}\sum_{i:y_i=k}(x_{it}-\bar{x}_t^k)(x_{ij}-\bar{x}_j^k)$. Using these definitions, the estimates (3.2.2) and (3.2.3) can be rewritten:

$$\begin{aligned}\hat{\mu}_t^k =& \frac{1}{n^k}\sum_{i:y_i=k} \boldsymbol{x}_i^\top \boldsymbol{u}_t \\ =& \left(\frac{1}{n^k}\sum_{i:y_i=k} \boldsymbol{x}_i\right)^\top \boldsymbol{u}_t \\ =& \left(\bar{\boldsymbol{x}}^k\right)^\top \boldsymbol{u}_t \end{aligned} \tag{3.2.6}$$

and

$$\begin{aligned}\hat{\sigma}_{tt}^k =& \frac{1}{n^k}\sum_{i:y_i=k}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t - \hat{\mu}_t^k\right)^2 \\ =& \frac{1}{n^k}\sum_{i:y_i=k}\left(\sum_{z=1}^p x_{iz}u_{tz} - \sum_{z=1}^p \bar{x}_z^k u_{tz}\right)^2 \\ =& \frac{1}{n^k}\sum_{i:y_i=k}\left(\sum_{z=1}^p \left(x_{iz}u_{tz} - \bar{x}_z^k u_{tz}\right)\right)^2 \\ =& \frac{1}{n^k}\sum_{i:y_i=k}\sum_{z=1}^p\sum_{j=1}^p u_{tz}u_{tj}\left(x_{iz}-\bar{x}_z^k\right)\left(x_{ij}-\bar{x}_j^k\right) \\ =& \sum_{z=1}^p\sum_{j=1}^p u_{tz}u_{tj}s_{zj}^k \\ =& \boldsymbol{u}_t^\top \boldsymbol{S}^k \boldsymbol{u}_t. \end{aligned} \tag{3.2.7}$$

In addition to these versions of $\hat{\mu}_t^k$ and $\hat{\sigma}_{tt}^k$, define: $\hat{\omega}_{iz}^k = \frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \hat{\mu}_z^k}{\sqrt{\hat{\sigma}_{zz}^k}}$. Then, using the product- and chain rule, it follows:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{v}_z}\hat{f}_z^k(\boldsymbol{x}_i^\top \boldsymbol{u}_z) =& \frac{1}{\sqrt{2\pi}}\frac{\partial}{\partial \boldsymbol{v}_z}\left(\frac{1}{\sqrt{\hat{\sigma}_{zz}^k}}\exp\left(-\frac{1}{2}\left(\hat{\omega}_{iz}^k\right)^2\right)\right) \\ =& \frac{1}{\sqrt{2\pi}}\left(\left(\frac{\partial}{\partial \boldsymbol{v}_z}\frac{1}{\sqrt{\hat{\sigma}_{zz}^k}}\right)\exp\left(-\frac{1}{2}\left(\hat{\omega}_{iz}^k\right)^2\right)\right. \\ &\left. + \frac{1}{\sqrt{\hat{\sigma}_{zz}^k}}\left(\frac{\partial}{\partial \boldsymbol{v}_z}\exp\left(-\frac{1}{2}\left(\hat{\omega}_{iz}^k\right)^2\right)\right)\right), \end{aligned} \tag{3.2.8}$$

which comes down to requiring:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{v}_z}\hat{\sigma}_{zz}^k =& \frac{\partial}{\partial \boldsymbol{v}_z}\boldsymbol{u}_z^\top \boldsymbol{S}^k \boldsymbol{u}_z \\ =& \frac{\partial}{\partial \boldsymbol{v}_z}\boldsymbol{v}_z^\top \boldsymbol{S}^k \boldsymbol{v}_z \left\|\boldsymbol{v}_z\right\|^{-2} \\ =& \left(\frac{\partial}{\partial \boldsymbol{v}_z}\boldsymbol{v}_z^\top \boldsymbol{S}^k \boldsymbol{v}_z\right)\left\|\boldsymbol{v}_z\right\|^{-2} + \boldsymbol{v}_z^\top \boldsymbol{S}^k \boldsymbol{v}_z\left(\frac{\partial}{\partial \boldsymbol{v}_z}\left\|\boldsymbol{v}_z\right\|^{-2}\right) \\ =& \frac{2}{\left\|\boldsymbol{v}_z\right\|}\left(\boldsymbol{S}^k - \hat{\sigma}_{zz}^k \boldsymbol{I}_p\right)\boldsymbol{u}_z, \end{aligned} \tag{3.2.9}$$

where $\boldsymbol{I}_p$ denotes the $p \times p$ identity matrix, and

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{v}_z}\hat{\omega}_{iz}^k =& \frac{\partial}{\partial \boldsymbol{v}_z}\frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \hat{\mu}_z^k}{\sqrt{\hat{\sigma}_{zz}^k}} \\ =& \frac{\partial}{\partial \boldsymbol{v}_z}\frac{\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^k\right)^\top \boldsymbol{u}_z}{\sqrt{\hat{\sigma}_{zz}^k}} \\ =& \frac{\left(\frac{\partial}{\partial \boldsymbol{v}_z}\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^k\right)^\top \boldsymbol{u}_z\right)\sqrt{\hat{\sigma}_{zz}^k} - \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^k\right)^\top \boldsymbol{u}_z\left(\frac{\partial}{\partial \boldsymbol{v}_z}\sqrt{\hat{\sigma}_{zz}^k}\right)}{\hat{\sigma}_{zz}^k} \\ =& \frac{1}{\sqrt{\hat{\sigma}_{zz}^k}\,\|\boldsymbol{v}_z\|}\left(\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^k\right) - \frac{\hat{\omega}_{iz}^k}{\sqrt{\hat{\sigma}_{zz}^k}}\boldsymbol{S}^k\boldsymbol{u}_z\right). \end{aligned} \tag{3.2.10}$$

Finally, combining (3.2.8) with (3.2.9) and (3.2.10) gives:

$$\frac{\partial}{\partial \boldsymbol{v}_z}\hat{f}_z^k(\boldsymbol{x}_i^\top \boldsymbol{u}_z) = -\frac{\exp\left(-\frac{1}{2}\left(\hat{\omega}_{iz}^k\right)^2\right)}{\sqrt{2\pi\hat{\sigma}_{zz}^k}\,\|\boldsymbol{v}_z\|}\left(\left(\frac{1}{\hat{\sigma}_{zz}^k}\boldsymbol{S}^k - \boldsymbol{I}_p\right)\boldsymbol{u}_z + \frac{\hat{\omega}_{iz}^k}{\sqrt{\hat{\sigma}_{zz}^k}}\left(\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^k\right) - \frac{\hat{\omega}_{iz}^k}{\sqrt{\hat{\sigma}_{zz}^k}}\boldsymbol{S}^k\boldsymbol{u}_z\right)\right),$$

which can then be substituted into (3.2.5). This, together with an appropriate starting value, say $\boldsymbol{V}_0$, is then presented to the BFGS algorithm which proceeds according to the discussion in Section 2.3. The choosing of $\boldsymbol{V}_0$ will be discussed in Section 3.5.1.

3.3 Projected Kernel Naïve Bayes

This section provides a nonparametric extension to the learning of a naïve Bayes improving projection matrix. It proceeds in the same order as the previous section, starting with a formulation of how f_t^k is estimated and followed by the computation of the vector gradients, $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}}$. Exchanging the Gaussian treatment for the unknown density function of the conditional random variable, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$, for kernel density estimation avoids imposing a potentially inappropriate parametric structure. The assumption of Gaussianity in (3.2.1) is not expected to hold for all data-sets. For those for which it does not, a KDE approach can offer improved classification accuracy. This follows since, in such cases, KDE could allow for a better approximation of the unknown density functions $\{f_t^k : t = 1, 2, \ldots, p'\}$ for $k = 1, 2, \ldots, K$. Furthermore, following the findings from John and Langley (1995), on data-sets where (3.2.1) does approximately hold, the KDE would learn this, and so, at least asymptotically, would not cause classification accuracy to deteriorate from that under Gaussianity. Building on the discussion of the bias-variance trade-off (see Section 1.3.1, and Section 3.5.2 below), a nonparametric treatment of the unknown density function, f_t^k, provides freedom in addition to that due to the projection matrix, $\boldsymbol{V}$. The classifier developed in this section is appropriately named Projected Kernel NB.

3.3.1 Posterior Treatment

From the sample data, $\boldsymbol{X}$, we can construct n^k transformed observations, say $\{\boldsymbol{x}_i^\top \boldsymbol{u}_t : i \in \{s : y_s = k\}\}$, for each of the random variables, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$. Then, proceeding according to (2.1.2), we have:

$$\hat{f}_t^k\left(\boldsymbol{x}^\top \boldsymbol{u}_t\right) = \begin{cases} \frac{1}{n^k h^k}\sum\limits_{\substack{s:y_s=k \\ s\neq i}} K\left(\frac{\boldsymbol{x}^\top \boldsymbol{u}_t - \boldsymbol{x}_s^\top \boldsymbol{u}_t}{h^k}\right), & \text{if } \boldsymbol{x} = \boldsymbol{x}_i \text{ with } y_i = k \\ \frac{1}{n^k h^k}\sum\limits_{s:y_s=k} K\left(\frac{\boldsymbol{x}^\top \boldsymbol{u}_t - \boldsymbol{x}_s^\top \boldsymbol{u}_t}{h^k}\right), & \text{otherwise,} \end{cases} \tag{3.3.1}$$

which is the KDE of the random variable, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$, at the point $\boldsymbol{x}^\top \boldsymbol{u}_t \in \mathbb{R}$. (Note that (3.3.1) is the "leave-one-out" variant of the KDE (Yao & Zhao, 2013), frequently used in projection pursuit, which decreases the risk of over-fitting). We will take the kernel function, $K(\cdot)$, as in (2.1.4). Section 2.1.2 gave a discussion on the computational efficiency allowed by this choice of kernel which, as can be seen in Figure 2.2, is shown to converge to the popular Gaussian kernel. The choice of bandwidth, h^k, will be discussed in Section 3.3.3 below. Also, recall that $n^k = \sum_{i=1}^n \mathbb{1}(y_i = k)$, the number of sample observations in each class. Replacing $\hat{f}_t^k$ in (3.2.4) with (3.3.1) specifies the optimisation problem under KDE. As before, it is not possible to analytically solve for the matrix, say $\boldsymbol{V}^*$, where $\ell(\boldsymbol{U}) \leq \ell(\boldsymbol{U}^*)$ (with $\boldsymbol{U} = \begin{bmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_{p'} \end{bmatrix}$ and $\boldsymbol{u}_t = \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}$). We again rely on the numerical optimisation strategy discussed in Section 2.3.

3.3.2 Optimisation: BFGS

It was noted that the Broyden, Fletcher, Goldfarb and Shanno algorithm approximates the inverse of the Hessian, $\frac{\partial^2 \ell(\boldsymbol{U})}{\partial \boldsymbol{V}^2}$, but that it still requires exact values of the first derivatives, $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}}$. To this end, we rely on the chain rule:

$$\begin{aligned}\left(\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_z}\right)^\top &= \left(\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{X}\boldsymbol{u}_z}\right)^\top \frac{\partial \boldsymbol{X}\boldsymbol{u}_z}{\partial \boldsymbol{v}_z} \\ &= \left(\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{X}\boldsymbol{u}_z}\right)^\top \frac{1}{\|\boldsymbol{v}_z\|} \boldsymbol{X}\left(\boldsymbol{I}_p - \boldsymbol{u}_z \boldsymbol{u}_z^\top\right), \end{aligned} \tag{3.3.2}$$

where

$$\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{X}\boldsymbol{u}_z} = \begin{bmatrix} \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{x}_1^\top \boldsymbol{u}_z} \\ \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{x}_2^\top \boldsymbol{u}_z} \\ \vdots \\ \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{x}_n^\top \boldsymbol{u}_z} \end{bmatrix}. \tag{3.3.3}$$

Then, from looking at $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{V}} = \begin{bmatrix} \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_1} & \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_2} & \dots & \frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{v}_{p'}} \end{bmatrix}$, with each column denoting a vector gradient, our main concern is with finding $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}$ for each of $j = 1, 2, \dots, n$ and $z = 1, 2, \dots, p'$.

Computation of $\frac{\partial \ell(\boldsymbol{U})}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}$

We require (with $i \neq j$ and where K' denotes the first derivative of K):

$$\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right) = \frac{1}{n^{y_j}\left(h^{y_j}\right)^2} \sum_{\substack{s: y_s = y_j \\ s \neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_s^\top \boldsymbol{u}_z}{h^{y_j}}\right), \tag{3.3.4}$$

$$\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \hat{f}_z^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right) = \frac{-1}{n^{y_j}\left(h^{y_j}\right)^2} K'\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \boldsymbol{x}_j^\top \boldsymbol{u}_z}{h^{y_j}}\right), \tag{3.3.5}$$

$$\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \hat{f}_z^{y_i}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right) = \frac{1}{n^{y_i}\left(h^{y_i}\right)^2} \sum_{\substack{s: y_s = y_i \\ s \neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_s^\top \boldsymbol{u}_z}{h^{y_i}}\right) \tag{3.3.6}$$

and

$$\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \hat{f}_z^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right) = \begin{cases} \frac{-1}{n^{y_i}(h^{y_i})^2} K'\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \boldsymbol{x}_j^\top \boldsymbol{u}_z}{h^{y_i}}\right), & y_i = y_j \\ 0, & \text{otherwise.} \end{cases} \tag{3.3.7}$$

Using (3.3.4), (3.3.5), (3.3.6) and (3.3.7), evaluation of the derivative of the first term in (3.2.4) yields:

$$\begin{aligned}
&\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \sum_{i=1}^{n} \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^{y_i}\right) \\
=&\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \sum_{i:y_i=y_j} \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^{y_i}\right) \\
=&\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_j}\right) + \sum_{\substack{i:y_i=y_j \\ i\neq j}} \frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_i}\right) \\
=&\frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\left(\prod_{t=1}^{p'} \hat{f}_t^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_j}}{\left(\prod_{t=1}^{p'} \hat{f}_t^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_j}} + \sum_{\substack{i:y_i=y_j \\ i\neq j}} \frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_i}}{\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_i}} \\
=&\frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)}{\hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)} + \sum_{\substack{i:y_i=y_j \\ i\neq j}} \frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\hat{f}_z^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right)}{\hat{f}_z^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right)} \\
=&\frac{\frac{1}{n^{y_j}\left(h^{y_j}\right)^2}\sum_{\substack{s:y_s=y_j \\ \ell\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_s^\top \boldsymbol{u}_z}{h^{y_j}}\right)}{\hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)} + \sum_{\substack{i:y_i=y_j \\ i\neq j}} \frac{\frac{-1}{n^{y_i}(h^{y_i})^2} K'\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \boldsymbol{x}_j^\top \boldsymbol{u}_z}{h^{y_i}}\right)}{\hat{f}_z^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right)} \\
=&\sum_{\substack{i:y_i=y_j \\ i\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_i^\top \boldsymbol{u}_z}{h^{y_j}}\right)\frac{1}{n^{y_j}\left(h^{y_j}\right)^2 \hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)} \\
&+ \sum_{\substack{i:y_i=y_j \\ i\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_i^\top \boldsymbol{u}_z}{h^{y_i}}\right)\frac{1}{n^{y_i}\left(h^{y_i}\right)^2 \hat{f}_z^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right)} \\
=&\sum_{\substack{i:y_i=y_j \\ i\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_i^\top \boldsymbol{u}_z}{h^{y_i}}\right)\left(\left(n^{y_j}\left(h^{y_j}\right)^2 \hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)\right)^{-1}\right. \\
&\left.+ \left(n^{y_j}\left(h^{y_j}\right)^2 \hat{f}_z^{y_j}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)\right)^{-1}\right)
\end{aligned} \tag{3.3.8}$$

Similarly, evaluation of the derivative of the second term in (3.2.4) gives:

$$\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^{k}\right)$$

$$
\begin{aligned}
&= \frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z} \log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k(\boldsymbol{x}_j^\top \boldsymbol{u}_t)\right)\hat{\pi}^k\right) + \frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\sum_{i\neq j}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^k\right) \\
&= \frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} + \sum_{i\neq j}\frac{\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&= \frac{\sum_{k=1}^{K}\left(\prod_{t\neq z} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k \frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\hat{f}_z^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_z\right)}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} + \sum_{i\neq j}\frac{\left(\prod_{t\neq z} \hat{f}_t^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_j}\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\hat{f}_z^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_z\right)}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&= \frac{\sum_{k=1}^{K}\left(\prod_{t\neq z} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k \frac{1}{n^k\left(h^k\right)^2}\sum_{\substack{s:y_s=k \\ s\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_s^\top \boldsymbol{u}_z}{h^k}\right)}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&\quad + \sum_{i\neq j}\frac{\left(\prod_{t\neq z} \hat{f}_t^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^{y_j}\frac{-1}{n^{y_j}\left(h^{y_j}\right)^2}K'\left(\frac{\boldsymbol{x}_i^\top \boldsymbol{u}_z - \boldsymbol{x}_j^\top \boldsymbol{u}_z}{h^{y_j}}\right)}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&= \sum_{k=1}^{K}\sum_{\substack{s:y_s=k \\ s\neq j}} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_s^\top \boldsymbol{u}_z}{h^k}\right)\frac{\left(\prod_{t\neq z} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\frac{\hat{\pi}^k}{n^k\left(h^k\right)^2}}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&\quad + \sum_{i\neq j} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_i^\top \boldsymbol{u}_z}{h^{y_j}}\right)\frac{\left(\prod_{t\neq z} \hat{f}_t^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\frac{\hat{\pi}^{y_j}}{n^{y_j}\left(h^{y_j}\right)^2}}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} \\
&= \sum_{i\neq j} K'\left(\frac{\boldsymbol{x}_j^\top \boldsymbol{u}_z - \boldsymbol{x}_i^\top \boldsymbol{u}_z}{h^{y_j}}\right)\left(\frac{\left(\prod_{t\neq z} \hat{f}_t^{y_i}\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\frac{1}{n(h^{y_i})^2}}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_j^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k} + \frac{\left(\prod_{t\neq z} \hat{f}_t^{y_j}\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\frac{1}{n\left(h^{y_j}\right)^2}}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{x}_i^\top \boldsymbol{u}_t\right)\right)\hat{\pi}^k}\right). \qquad (3.3.9)
\end{aligned}
$$

Looking back to (2.1.3), it is seen that both derivatives, $\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\sum_{i=1}^{n}\log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^{y_i}\right)$ and $\frac{\partial}{\partial \boldsymbol{x}_j^\top \boldsymbol{u}_z}\sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^k(\boldsymbol{x}_i^\top \boldsymbol{u}_t)\right)\hat{\pi}^k\right)$, take the form of the sum of a weighted kernel function: replacing $K(\cdot)$ in (2.1.11) with $K'(\cdot)$, and choosing w_i appropriately for each of (3.3.8) and (3.3.9). This is exactly what is needed for the fast computation shown in Section 2.1.2. Combining (3.3.8) and (3.3.9) can be used for the computation of (3.3.3), which can then be substituted into (3.3.2) – the vector gradient. As before, this, together with an appropriate starting value, say $\boldsymbol{V}_0$, is then presented to the BFGS algorithm.

3.3.3 Choosing the Bandwidth

In Section 2.1.1 we discussed that the bandwidth determines the smoothness of the KDE. Small values of h^k correspond to a wiggly curve that is too local (with the resulting estimator assigning too much weight to the observations in close vicinity to the point being evaluated, and discounting the contributions of those further away excessively). Too large values of bandwidth enacts oversmoothing, moving the estimator toward assigning a near identical density estimate over the entire range of the data (consult Figure 2.1). We noted that the optimal choice of bandwidth (in terms of density estimation) depends on the smoothness of the underlying target density, but that, as this smoothness is unknown, the difficulty with KDE is in deciding on a sensible bandwidth estimation approach.

The choice of h^k in this thesis is a modified version of Silverman's rule-of-thumb (1986). Said rule is based on minimising the asymptotic mean integrated squared error when using the Gaussian kernel (and is optimal in this sense if the density being estimated is itself Guassian). We take:

$$h^k = \frac{0.1}{\left(n^k\right)^{\frac{1}{4+p'}}} \sqrt{\frac{1}{p'} \sum_{t=1}^{p'} \lambda_t^k} \qquad k = 1, 2, \ldots, K \tag{3.3.10}$$

where λ_t^k denotes the t^{th} eigenvalue of the sample covariance matrix, $\boldsymbol{S}^k$. This choice of bandwidth was found to work well. However, it is thought to be worthwhile for a future study to focus on improving our choice of bandwidth. For one thing, (3.3.10) remains fixed throughout the BFGS procedure. A sensible improvement would be one that updates h^k as the optimisation of the projection continues. Another option is to choose the bandwidth in a supervised way. That is, using the information available in the labels, $\boldsymbol{y}$, that accompany $\boldsymbol{X}$. This follows from the observation that the choice of bandwidth delivering the optimal rate of convergence of $\hat{f}_t^k$ to the target density, f_t^k, does not necessarily agree with the optimal choice in a classification accuracy sense.

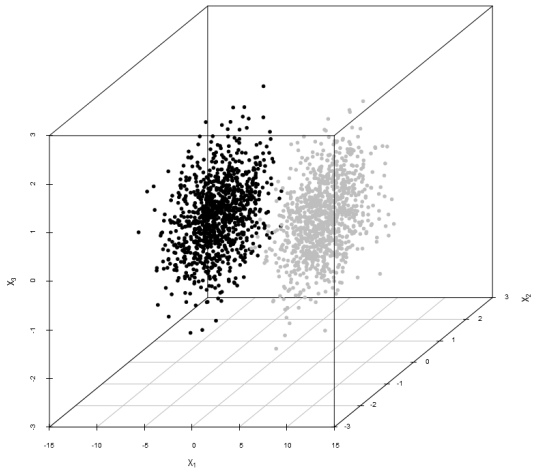
3.4 Connection to Independent Component Analysis

A comparison of the objective functions due to independent component analysis and the likelihood-based learning of a projection matrix proposed in this chapter reveals the similarity between the two procedures. In this section we show the latter of these two objectives to be an unconstrained penalised version of the former. The effect of said penalty is argued to be two-fold in that it encourages solutions that are both orthogonal and that favour classification accuracy under naïve Bayes.

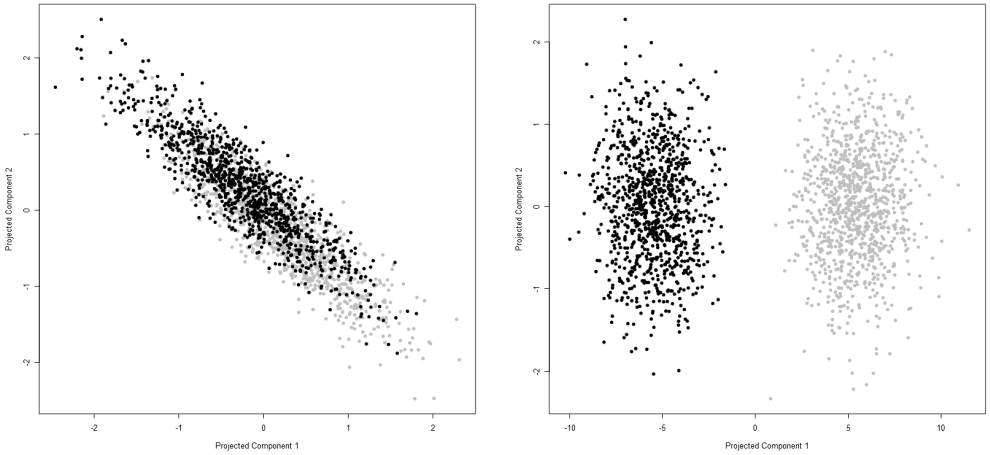
Recall that the goal of ICA is to recover latent sources which are assumed to only be observable indirectly. This task can be framed in terms of finding an unmixing matrix, say $\boldsymbol{B}$, that maximises the statistical independence of the components of the random vector, $\mathbf{S} = \boldsymbol{B}\mathbf{X}$. Expanding the ICA objective, (2.2.8), follows as:

$$\begin{aligned} I(\mathbf{S}) &= \sum_{j=1}^{p'} H\left(\mathrm{S}_j\right) + c \\ &= -\sum_{j=1}^{p'} \mathrm{E}\left(\log\left(f_{\mathrm{S}_j}\left(\mathrm{S}_j\right)\right)\right) + c \\ &= -\sum_{j=1}^{p'} \mathrm{E}\left(\log\left(f_{\mathbf{X}^\top \boldsymbol{b}_j}\left(\mathbf{X}^\top \boldsymbol{b}_j\right)\right)\right) + c \end{aligned}$$

where $c \in \mathbb{R}$ is some constant which does not depend on the unmixing matrix, and where $f_{\mathbf{X}^\top \boldsymbol{b}_j}$ denotes the density function of the random variable, $\mathbf{X}^\top \boldsymbol{b}_j$. The function $H(\cdot)$ denotes



(a) Gaussian mixture: original representation.



(b) PKNB representation without penalty.

(c) PKNB representation.

Figure 3.1: Illustration: effect of penalty term in (3.4.2).

the entropy of a stochastic variable – a quantity which was introduced in Section 2.2.1 as a measure of a variable's "randomness". Noting that minimising $I(\mathbf{S})$ is equivalent to maximising the negative of this quantity, substituting an estimate for $\mathrm{E}\left(\log\left(f_{\mathbf{X}^\top \boldsymbol{b}_j}\left(\mathbf{X}^\top \boldsymbol{b}_j\right)\right)\right)$ into the above (while discarding the constant term), the ICA task can be framed as the pursuit of the unmixing matrix maximising:

$$\sum_{i=1}^{n}\sum_{j=1}^{p'}\log\left(\hat{f}_{\mathbf{X}^\top \boldsymbol{b}_j}\left(\boldsymbol{x}_i^\top \boldsymbol{b}_j\right)\right) \qquad \text{such that } \boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}_p. \tag{3.4.1}$$

A natural candidate to consider when faced with the unknown density function, $f_{\mathbf{X}^\top \boldsymbol{b}_j}$, is the nonparametric kernel density estimator. This avoids imposing some potentially inappropriate parametric structure, and allows for flexible estimation of the unknown density functions. In addition, Hofmeyr's (2019a) fast variation of KDE means that the unmixing matrix maximising (3.3.1) can be computed efficiently.

The objective function due to the likelihood-based learning of a projection matrix for improving NB (i.e., that due to our method) follows from (3.2.4) as:

$$\begin{aligned}\ell(\boldsymbol{V}) &= \sum_{i=1}^{n}\log\left(\left(\prod_{t=1}^{p'}\hat{f}_t^{y_i}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{y_i}\right) - \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'}\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{k}\right)\\ &= \sum_{i=1}^{n}\sum_{t=1}^{p'}\log\left(\hat{f}_t^{y_i}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right) + \sum_{i=1}^{n}\log\left(\hat{\pi}^{y_i}\right) - \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'}\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{k}\right)\end{aligned}$$

under the constraint that $\boldsymbol{V}$ is such that each of its columns have unit norm. As the second term here does not depend on the projection matrix, we seek to find the matrix maximising:

$$\sum_{i=1}^{n}\sum_{t=1}^{p'}\log\left(\hat{f}_t^{y_i}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right) - \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'}\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{k}\right) \tag{3.4.2}$$

$$= \sum_{k=1}^{K}\sum_{i:y_i=k}\sum_{t=1}^{p'}\log\left(\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right) - \sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'}\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{k}\right) \tag{3.4.3}$$

recalling that $\hat{f}_t^k$ denotes the estimated density function of the conditional random variable, $\mathbf{X}^\top\boldsymbol{v}_t \mid \mathrm{Y} = k$. This is done, in Section 3.2, according to (3.2.1), and is taken as (3.3.1) in Section 3.3. Notice, $\boldsymbol{u}_t = \boldsymbol{v}_t$ under the assumption of the columns of $\boldsymbol{V}$ having unit norm (i.e., $\boldsymbol{U} = \boldsymbol{V}$).

A comparison of (3.4.1) and (3.4.2) reveals the similarity in learning the unmixing matrix, $\boldsymbol{B}$, due to ICA, and the projection matrix, $\boldsymbol{V}$, proposed to improve the classification accuracy of NB. Both objective functions are of the form of a sum of the log of density estimates. However, an inspection discloses that the objective due to our projection matrix differs from that of ICA in two distinct ways. Ignoring the case of Section 3.2 estimating the unknown densities parametrically, and, for now, the orthogonality constraint in (3.4.1), the first difference is (3.4.2) approaching said densities in a supervised way. That is, treating each of the K classes according to its own density estimate. With the *unconditional* independence outcome of ICA, (3.4.1), in mind, the supervision is argued to effect *conditional* independence. The sensibility of this amendment can be motivated from the discussion of Simpson's paradox in Section 1.4. There we saw that unconditional independence does not guarantee the conditional type (that to which the success of NB is related) i.e., that the components of $\boldsymbol{V}^\top\mathbf{X}$ being statistically independent does not imply the same for the components of $\{\boldsymbol{V}^\top\mathbf{X} \mid \mathrm{Y} = k : k = 1, 2, \ldots, K\}$. The other difference is the additional term of (3.4.2). Recalling that we wish to maximise the above objective functions, the second term, $\sum_{i=1}^{n}\log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'}\hat{f}_t^{k}(\boldsymbol{x}_i^\top\boldsymbol{v}_t)\right)\hat{\pi}^{k}\right)$, which can be seen as a penalty, enforces regularisation (in the sense that it induces a particular outcome, and so regularises the outcome by making it less variable). Figure 3.1 visualises the effect of this penalty. The projected representation, $\boldsymbol{XV}$, of Guassian mixture data (the top panel) that maximises (3.4.2) with and without the penalty term is compared. From this we gather that the impact of the penalty is two-fold. The inclusion of the second term in (3.4.2) discourages those projections that do not allow for class discrimination via density functions. This is seen from the lack of class overlap along the first projected component in the PKNB representation (which is not the case when learning the projection without the penalty term). In addition, comparing the orientation of the projected data in the bottom panels of Figure 3.1, we see that the penalty term also plays a role similar to the orthogonality constraint of (3.4.1). By discouraging non-orthogonal solutions, together with the ICA-type first term of (3.4.2), the penalty effects CC independence. Both these effects are expected to deliver improved classification accuracy for NB.

Section 2.2.4 concluded with a direct application of ICA to NB – the CC ICA due to Bressan and Vitrià (2002). This involves, following from (3.4.1), learning K separate unmixing

matrices, $\boldsymbol{B}^k$, which can be performed according to:

$$\sum_{i:y_i=k} \sum_{j=1}^{p'} \log\left(\hat{f}^k_{\boldsymbol{X}^\top \boldsymbol{b}^k_j}\left(\boldsymbol{x}_i^\top \boldsymbol{b}^k_j\right)\right) \qquad \text{such that } \left(\boldsymbol{B}^k\right)^\top \boldsymbol{B}^k = \boldsymbol{I}_p, \tag{3.4.4}$$

for each of $k = 1, 2, \ldots, K$. Comparing (3.4.3) to (3.4.4) reinforces the sensibility of Projected NB. It appears that our method attempts the CC independence of CC ICA via a single, rather than K separate, projections. This is seen from the first term of (3.4.3) being similar to that of (3.4.4). This term, together with the (approximate) orthogonal effect of the penalty, preserves the *conditional* independence of CC ICA. Furthermore, as argued above, the penalty term of Projected NB also discourages solutions that do not allow for class discrimination via density functions. Therefore, compared to CC ICA, our method is both more parsimonious and is explicitly formulated to favour solutions that allow for class discrimination. Note that (3.4.4) differs from the FastICA algorithm (which seeks to directly maximise non-Gaussianity) used by the authors in the original paper (see Hyvärinen (1999) for a thorough discussion of FastICA).

3.5 Practical Matters

This section looks at the practical aspects of the proposed naïve Bayes improvement. It briefly discusses the issue of a suitable initialisation of the projection matrix to be passed to the BFGS algorithm, and also looks into how to go about choosing p'. That is, choosing of the size of the projection matrix, $\boldsymbol{V}$.

3.5.1 Initialisation of the Projection Matrix

The discussion in Section 2.3.1 suggested that the starting point of a numerical procedure weighs on its ability to reach optimality. A poor choice of $\boldsymbol{V}_0$ would make getting stuck at some local optimum more likely, or rather, would make finding the projection maximising (3.2.4) less likely. Serious attention and a thorough study, however, of different initialisation practices was not part of the focus of this thesis. Instead, borrowing from linear discriminant analysis (LDA), the columns of $\boldsymbol{V}_0$ are initialised to be the eigenvectors corresponding to the p' largest eigenvalues of:

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b \tag{3.5.1}$$

where $\boldsymbol{S}_w = \sum_{i=1}^n \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{y_i}\right)\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}^{y_i}\right)^\top$ is the "within classes scatter matrix" and, taking $\bar{\boldsymbol{x}} = \frac{1}{n^k}\sum_{k=1}^K \bar{\boldsymbol{x}}^k$ as the sample mean vector over all classes, $\boldsymbol{S}_b = \sum_{i=1}^n \left(\bar{\boldsymbol{x}}^{y_i} - \bar{\boldsymbol{x}}\right)\left(\bar{\boldsymbol{x}}^{y_i} - \bar{\boldsymbol{x}}\right)^\top$ the "between classes scatter matrix". Keeping in mind that the rank of (3.5.1) is $K-1$, in cases where $p' > K-1$, (3.5.1) is slightly amended and the columns of $\boldsymbol{V}_0$ are initialised to be the eigenvectors corresponding to the p' largest eigenvalues of:

$$\boldsymbol{S}_w^{-1}\boldsymbol{S}_b + \epsilon\boldsymbol{S} \tag{3.5.2}$$

where $\epsilon > 0$ and where $\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^n \left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)\left(\boldsymbol{x}_i - \bar{\boldsymbol{x}}\right)^\top$ is the estimated total covariance matrix. This amendment means that, when ϵ is small, the first $K-1$ columns of $\boldsymbol{V}_0$ will be close to those due to (3.5.1). The remaining $p' - (K-1)$ columns will be dominated by the second term in (3.5.2), which will be such that the data projected onto them will have high variance.

The described initialisation means that the search for the optimal projection matrix starts with reasonable separation of the classes (albeit, a linear separation being favoured), which is then improved by the BFGS algorithm that refines the initial projection according to the objectives discussed in Sections 3.2.2 and 3.3.2.

3.5.2 Size of the Projection Matrix

The Bias-Variance Trade-Off

Section 1.3.1 made the case that the success of NB can be explained according to the low variance implication of its assumption of class conditional mutual statistical independence. It was said that this assumption, in effect, trades a variance reduction for bias in estimating the posterior probability, $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$. However, the particular combination of bias and variance due to the standard NB (that described in Section 1.2) might not be optimal in a classification accuracy sense, and so it was suggested in Section 3.1.1 that a sensible approach to improving NB would be one that traverses the bias-variance trade-off to a more favourable position. Viewing the entries in the projection matrix, $\boldsymbol{V}$, as parameters, our proposed method can be seen to do exactly this.

A comparison of the standard Gaussian NB and the projection-based classifier proposed in Section 3.2, Projected Gaussian NB, reveals the difference in its independence treatment. The former, for each of the K classes, assumes the predictors, $\mathrm{X}_j \mid \mathrm{Y} = k \sim \mathrm{N}(\mu_j^k, \sigma_{jj}^k)$, are class conditionally mutually statistically independent for $j = 1, 2, \ldots, p$, whereas the latter assumes independence for the projected predictors, $\mathbf{X}^\top \boldsymbol{v}_t \mid \mathrm{Y} = k \sim \mathrm{N}\left(\mu_t^k, \sigma_{tt}^k\right)$, for $t = 1, 2, \ldots, p'$ (following Section 3.4, the projection is adapted to effect this CC independence). Now, ignoring the $\mathrm{P}(\mathrm{Y} = k)$ term (i.e., the class priors), the standard Gaussian NB requires a total of $2Kp$ parameters to model $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$, whereas the projection-based model needs $pp' + 2Kp' - p'$ (accounting for the impact of restricting each of the columns of $\boldsymbol{V}$ to have unit ℓ_2-norm). From this, and keeping in mind that $p' \leq p$ (the number of predictors), it follows that PGNB exceeds the parameter count of the standard case when:

$$p' > \frac{2Kp}{2K + p - 1}. \tag{3.5.3}$$

Additional parameters suggests closer modeling of the (true) underlying probability, but also implies more quantities that need to be estimated from the sample data, $(\boldsymbol{X}, \boldsymbol{y})$. Thus, when (3.5.3) is the case, we expect a more flexible estimator. That is, we expect the estimator of the posterior probability due to PGNB to be less biased, but also to be more variable than the estimator due to the standard Gaussian NB. Similarly, when the reverse holds (i.e., $p' < \frac{2Kp}{2K+p-1}$), where the opposite is likely true: we expect the estimator of $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$ to be more biased and less variable. A key specification, therefore, needed for the learning of $\boldsymbol{V}$ is p' – the size of the projection matrix.

The further p' deviates from $\frac{2Kp}{2K+p-1}$, the bigger the expected departure from the standard Gaussian NB. However, even when PGNB has the same number of degrees of freedom as the standard Gaussian model (i.e., $p' = \frac{2Kp}{2K+p-1}$), we expect the former of these to utilise said freedom more effectively. This is based on the notion that two models being equally flexible can differ vastly and, hence, deliver vastly different classification performance. It is our feeling that the formulation of PGNB is superior, and so all things being equal, it should outperform the standard Gaussian NB.

We noted in the introduction to Section 3.3 that the nonparametric extension provides freedom in addition to that offered by the projection. Comparing the degrees of freedom of the semiparametric classifier, Projected Kernel NB, due to Section 3.3 to the standard Gaussian NB or to PGNB is difficult: we cannot state a condition such as (3.5.3) for when the PKNB is more flexible than the other two cases. However, we do note that the effect of swapping Gaussianity for KDE is expected to be a bias reduction and variance increase for the estimator of the relevant posterior probability (the extent of which being controlled by the bandwidth). We also expect the PKNB to utilise its degrees of freedom more effectively than the standard kernel NB model.

Dimension Reduction

Any choice of p' such that $p' < p$ implies a dimension reduction of the original input. Beyond potential visualisation benefits (see below), transforming $\boldsymbol{X}$ to a lower dimensional space could deal with collinearities in the data and it may well be that the representation that best matches the NB assumption is found in some lower dimensional subspace. However, choosing p' to be too small would cause the transformed predictors to lose their discriminatory power. Hence, in addition to fewer parameters implying more biased estimates of the posterior probability, there is danger in choosing too small a size for the projection matrix in that it could effect a loss of discriminatory capacity for the set of transformed (or, projected) predictors, $\boldsymbol{V}^\top \mathbf{X}$. Taking too large a value for p' is also dangerous as it implies a large number of parameters to be learnt. This risks overfitting (Hastie *et al.*, 2008:219-223). Section 1.2 proposed that supervised learning is concerned with discovering regularity in sample data, and that, once this regularity is established, it can be applied to unlabeled-, or "test", cases. The success of a classifier on the sample data is termed its training performance, and its success on test cases, that in which we are interested in, is referred to as its generalisation performance. When the sample data are overfitted it is modeled too closely. The regularity of the sample data is, in a sense, "faked", and confused with patterns which invariably arise in any realisation of stochastic data, but which are only attributable to randomness and not to any actual regularity. Overfitting corresponds to poor generalisation performance, and so its risk should be considered when choosing p'.

Visualisation

The projection matrix, $\boldsymbol{V}$, is chosen according to that maximising the likelihood under NB. Recalling the link between likelihood maximisation and classification accuracy, this tends to translate into class separation. Similar to what can be done with the LDA coefficients discussed in Section 3.5.1, viewed in two- or three dimensional space (i.e., looking at $\boldsymbol{XV}$ where p' is taken equal to 2 or 3), the projection should deliver improved visual class distinction over that of the original representation. That is, the projected data are expected to be class conditionally clustered. In addition, then, to being easier to interpret and apply, the projected NB has a visualisation edge over class conditional independent component analysis. This observation builds on Section 3.4, which ended with the point that our method has the advantage of being more parsimonious than CC ICA since it requires only a single projection. Importantly, this visualisation benefit is not possible under CC ICA, which learns each unmixing matrix individually. This means that a simultaneous inspection of the learnt effect over all classes is not generally possible (visualising a single class is not likely to be informative from the point of view of classification as there is no reason to expect that such a projection is going to deliver discrimination of classes). Unlike the LDA coefficients, though, we have no prior feeling as to the ordering of the (visual) discriminatory capacity of the different columns of the projection matrix. In Chapter 4, we compare the visualisation due to LDA with that of Projected NB when taking $p' = 2$.

Choosing p'

The choice of p' can be viewed as the challenge of balancing the goals of discriminatory capacity (that which deteriorates with too small a choice), and not overfitting the available data (the case when p' is too large, and implying poor generalisation performance). From the argument of Section 1.3.1, the standard Gaussian NB is potentially too biased, and so a choice of p' such that (3.5.3) holds is expected to be superior (at least for the case of the classifier due to Section 3.2). This follows since it allows the structural bias to be addressed. However, it may also be that smaller values of p' (i.e., where (3.5.3) does not hold) delivers a model with more discriminatory power (in the generalisation sense), and achieves superior classification accuracy (better than when (3.5.3) holds, or than the standard Gaussian NB), which can be explained from Projected NB making better use of its degrees of freedom. This, then, is the

luxury of p': control of the bias-variance trade-off in the context of a model with an otherwise rigid structure.

In the practical application of the method, p' can be selected via cross-validation, which chooses the value which minimises an estimate of the generalisation classification error rate. Section 4.1.2 also provides a rule-of-thumb when choosing p' according to what was found to work well with the experiments of Chapter 4.

3.6 Conclusion

This chapter detailed an improvement to the standard naïve Bayes classifier. This involved learning a transformed representation of the data via a likelihood-based objective function. Both a parametric and nonparametric version of Projected NB was given, Projected Gaussian NB and Projected Kernel NB. Considerable attention was paid to the computation of the relevant derivatives needed for optimisation using the BFGS algorithm. Following this, the connection between Projected NB and independent component analysis was established. Our method was argued as a modified version of class conditional ICA that preserves its *class conditional* effect, while being specifically tuned for classification accuracy and being more parsimonious. The final section attended to the initialisation of the projection matrix as well as to the matter of choosing its size. The impact of the latter was discussed in the context of the bias-variance trade-off and the benefits of a dimension reduction.

Chapter 4

Experimental Results

This chapter illustrates the practical application of the proposed naïve Bayes improvement. Those classifiers deemed similar to Projected NB in its motivation, or considered sensible to compare with, are briefly described. Then, we compare the performance of Projected NB to the selected competitors on both simulated and real-world data-sets. Our own simulation experiment, designed according to a mixture of Gaussian distributions, is presented. In addition, simulated data-sets from the popular R package "mlbench" (Leisch & Dimitriadou, 2015) are investigated. Real world data is obtained from the UCI Repository (Dua & Graff, 2019) and OpenML (Vanschoren, Van Rijn, Bischl & Torgo, 2013) website. The Bayes error rate and the Kullback-Leibler divergence are presented as sensible measures for the difficulty of a classification task. Attention to the visualisation capabilities of Projected NB is paid throughout this chapter. In particular, the success of Projected NB is compared to that of linear discriminant analysis in visualising selected real-world data.

4.1 Benchmarking Projected Naïve Bayes

This section briefly describes those models whose classification accuracy is compared to Projected naïve Bayes in the sequel. This comparison is performed to motivate our method as a significant improvement over standard NB, as well as to illustrate Projected NB to be competitive with the state-of-the-art. The section ends with a short discussion on how we measure classification accuracy, and on parameter tuning.

4.1.1 Competition Candidates

Standard Naïve Bayes (SGNB and SKNB)

The standard Gaussian naïve Bayes (SGNB) approximates the posterior probability, $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$, by assuming the random variables $\mathrm{X}_j \mid \mathrm{Y} = k$ to be statistically independent for $j = 1, 2, \ldots, p$ (i.e., (1.2.2)), and takes each of them to be Normally distributed with μ_j^k and σ_{jj}^k its mean and variance, respectively. These unknown parameters are estimated according to the maximum likelihood quantities, (1.2.7) and (1.2.8), and $\pi^k = \mathrm{P}(\mathrm{Y} = k)$ is estimated with the consistent estimator, (1.2.3). The details of SGNB can be found in Section 1.2.2. The standard kernel NB (SKNB) due to John and Langley (1995) works exactly as SGNB, but swaps Gaussianity for kernel density estimation. The kernel function in the sequel (for this method) is taken as the popular Gaussian kernel i.e.,

$$K(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right),$$

and the bandwidth is taken according to Silverman's rule-of-thumb (1986). SKNB is described in more detail in Sections 1.3.1 and 1.5. (Note, what is referred to as SKNB here was called "Flexible Bayes" by the originators of the method).

Class Conditional Independent Component Analysis Naïve Bayes (CCICA)

Using class conditional ICA together with NB involves learning K separate $p' \times p$ unmixing matrices, $\{\boldsymbol{B}^k : k = 1, 2, \ldots, K\}$. In accordance with Bressan and Vitrià (2002), this is done with the help of FastICA (Hyvärinen, 1999). Then, conditional on $\mathrm{Y} = k$, the components of the random vector, $\mathbf{S}^k = \boldsymbol{B}^k\mathbf{X}$, are approximately independent. With this independence and (1.2.2) in mind, and defining:

$$v^k = \left(\prod_{t=1}^{p} \lambda_t^k\right)^{-\frac{1}{2}},$$

where λ_t^k denotes the t^{th} eigenvalue of the sample covariance matrix, $\boldsymbol{S}^k$, classification is performed according to the class maximising:

$$\hat{\mathrm{P}}(\mathrm{Y} = k \mid \boldsymbol{x}) \propto v^k \left(\prod_{t=1}^{p'} \hat{f}_t^k\left(\boldsymbol{B}^k\boldsymbol{x}\right)\right) \hat{\pi}^k.$$

In the above, the joint density has been factorised as the product of marginal densities (see Section 1.4). Here, $\hat{f}_t^k$ denotes the kernel density estimate of the t^{th} component of the conditional random vector, $\mathbf{S}^k \mid \mathrm{Y} = k$. To this end, the Gaussian kernel and Silverman's rule-of-thumb are used. The prior class estimate, $\hat{\pi}^k$, is according to the consistent estimator, (1.2.3). The scaling constant, v^k, corrects for the impact of whitening the data. Choosing the unmixing matrices, $\boldsymbol{B}^k$, such that $p' < p$ delivered poor classification results. For this reason, p' was kept equal to the number of inputs.

Discriminant Analysis (LDA and QDA)

Linear discriminant analysis was proposed by Sir Ronald Fisher in *Annals of Eugenics* in 1936 (Anderson, 1996:30). Much like standard NB, the success of LDA can be explained from the low variance implication of its assumptions. LDA takes:

$$\mathbf{X} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}\right)$$

for each of the classes. This models the K conditional random vectors according to separate mean vectors, $\boldsymbol{\mu}^k$, while a common covariance matrix, $\boldsymbol{\Sigma}$, is assumed. The unknown parameters are approximated via maximum likelihood estimation, and $\pi^k = \mathrm{P}\left(\mathrm{Y} = k\right)$ is estimated with (1.2.3). Classification is performed by assigning input patterns to the class which maximises the posterior probability, $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$. Quadratic discriminant analysis (QDA) extends LDA by treating each conditional random vector according to its own covariance matrix:

$$\mathbf{X} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k\right).$$

This extension allows for the learning of nonlinear decision boundaries, and implies a more flexible treatment of $\mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x})$. LDA and QDA are sensible classifiers to compare with Projected NB. This follows from the flexibility of PGNB falling somewhere in between that of LDA and QDA.

Support Vector Machine (SVM)

The support vector machine (SVM) is due to Cortes and Vapnik (1995). There are many resources available on the subject. In particular, the reader is encouraged to consult Ng (2019). The SVM is a generalisation of the support vector classifier, which is itself a generalisation of the maximal margin classifier. In the context of a binary classification problem, the latter of these methods tries to directly fit a linear decision boundary via a so-called "separating

hyperplane". As the name suggests, it fits the hyperplane which maximises the distance (or, margin) to the closest observation from either class. This method suffers from only being applicable to data-sets that are linearly separable (that is, to those for which a separating hyperplane exists). The support vector classifier addresses this limitation. It also tries to fit a hyperplane with maximal margin, but permits a certain amount of slack by allowing observations to be on the "wrong" side of the margin or hyperplane. The SVM further extends the support vector classifier by fitting a nonlinear decision boundary. This is achieved by enlarging the input space via a kernel function, and then fitting a hyperplane in said enlarged space. The model is extended to multi-class (i.e., $K > 2$) classification tasks via the "One-Versus-One" or the "One-Versus-All" extension (the former of which is used in the sequel). SVMs require the specification of a kernel function together with a scale parameter, and a cost parameter (which controls the amount of slack). To this end the Gaussian kernel (known as the radial basis function kernel in the SVM literature) with the scale parameter set to $\frac{1}{p}$ is relied on. The cost parameter is put equal to 1.

Random Forest (RF)

Classification trees were formalised by Breiman, Friedman, Olshen and Stone (1984). A tree is a simple classifier that attempts to segregate the input space into homogeneous non-overlapping regions. Homogeneity can be measured in a number of ways including the Gini-index and cross-entropy. Segregation is done greedily via recursive binary splitting. In addition to being easy to understand, a classification tree is also easy to visualise and has the advantage of automatic variable selection. However, the classifier suffers from poor generalisation performance. It is a highly flexible method, and can be described as being of low bias but high variance. This weakness of instability can be addressed by an aggregation that combines the output of many trees. The success of doing so is due to the asymmetric influence the aggregation has on bias and variance (Hastie *et al.*, 2008:285). One such aggregation technique is the random forest (RF).

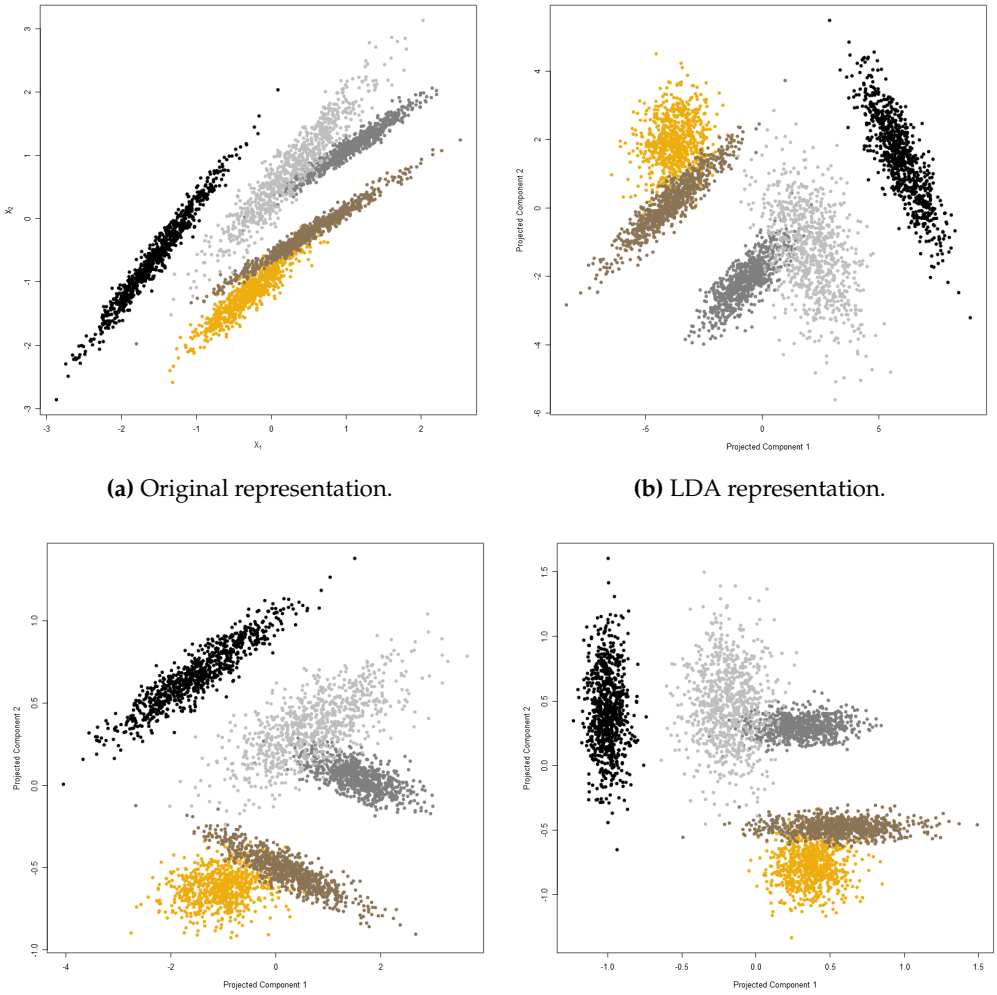
The RF is an amended version of bootstrap aggregation (or, "bagging") of classification trees. Bagging relies on bootstrap sampling available data. It fits a tree to each bootstrap sample, and aggregates the fitted trees according to a majority vote. The modification of the RF is that it only considers a random subset, of size m, of the available inputs at each step of the recursive binary splitting. This has the effect of decorrelating the constituent classification trees (James, Witten, Hastie and Tibshirani, 2013:319) which delivers a reduction in variance for the resulting ensemble classifier. The value m can be chosen via cross-validation. We simply take this value to be the closest integer to $\sqrt{p}$ – something which has been found by others to work well (Breiman, 2002).

4.1.2 Measuring Performance and Tuning Parameters

The error rate of a classifier on some set of observations, say $\{(\boldsymbol{x}_i, y_i) : i = 1, 2, \ldots, n\}$, is defined as:

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left(\hat{y}_i \neq y_i\right), \tag{4.1.1}$$

where $\hat{y}_i$ represents the predicted class of the input pattern, $\boldsymbol{x}_i$ (recall that $\mathbb{1}(\cdot)$ denotes the indicator function). Following Section 4.1.1, the scope for parameter tuning is limited to the choice of p' for PGNB and PKNB. As a rule-of-thumb, setting p' equal to the closest integer to $K\log(p)$ was found to work well (keeping in mind the restriction, $p' \leq p$) for both methods. Although a more extensive effort would involve choosing p' according to the sample data (e.g., via a measure such as cross-validation that seeks the parameter which minimises an estimate of the generalisation error rate), computational considerations meant that such an approach was too expensive. It is also noted that since the parameter settings for the other methods considered are kept fixed, it would be an unfair comparison to not do the same with our method. The limited amount of real-world data in Section 4.3 means that a search for the optimal p' would imply a difficult trade-off regarding the allocation of the available data.



(a) Original representation.

(b) LDA representation.

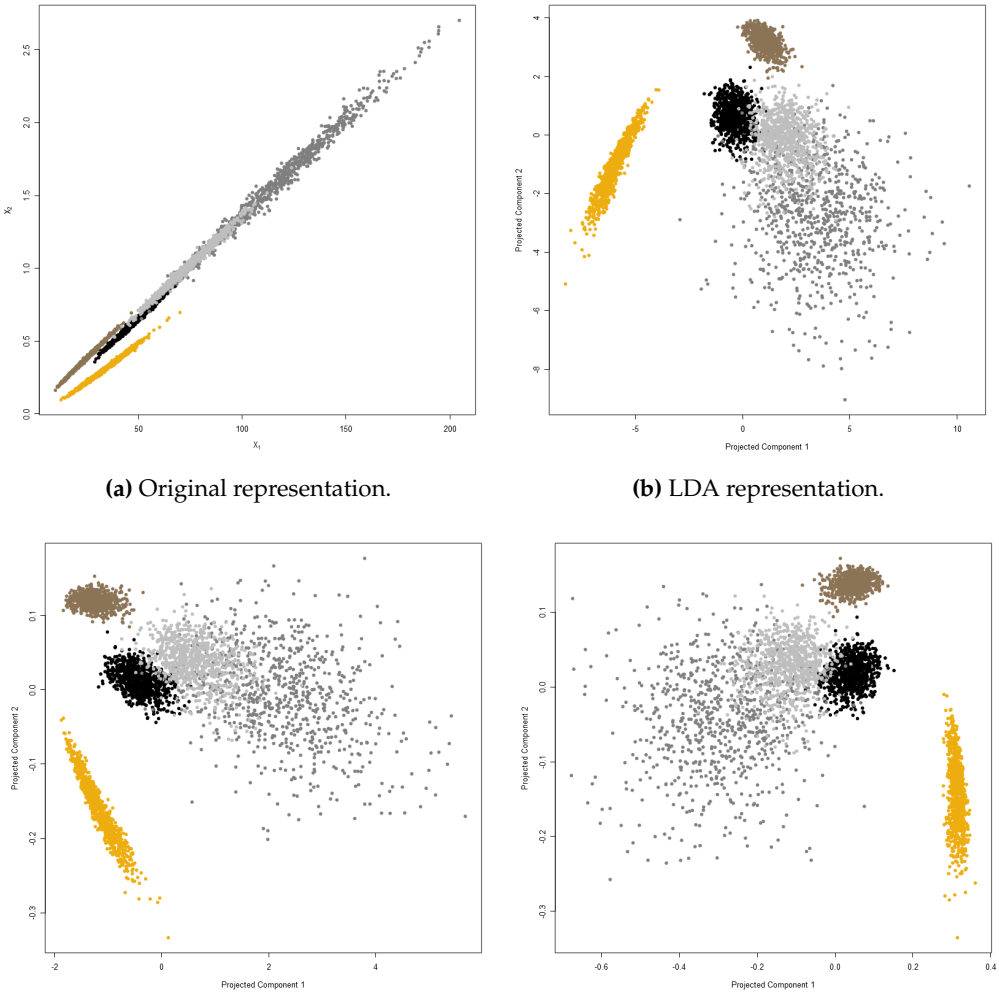
(c) PCA representation.

(d) PGNB representation.

Figure 4.1: Visual illustration of Gaussian mixture data with $p = 2$, $K = 5$, $C = 10$.

4.2 Simulation Study

This section illustrates the performance of the proposed naïve Bayes improvement via a simulation study. An experiment designed according to a mixture of Gaussian distributions is presented. Following this, the Spirals and XOR data-sets from the R package "mlbench" (Leisch & Dimitriadou, 2015) are investigated. The performance of Projected NB is compared to those competitors discussed in Section 4.1.1.



(a) Original representation.

(b) LDA representation.

(c) PCA representation.

(d) PKNB representation.

Figure 4.2: Visual illustration of squared Gaussian mixture data with $p = 2$, $K = 5, C = 25$.

4.2.1 Gaussian Mixture

In order to assess the ability of our proposed method to identify projections of the data which allow for accurate classification using NB, even in scenarios where the standard NB is inappropriate, data are generated in deliberate violation of the NB assumption of class conditional mutual statistical independence. Following the notation of Section 1.2.1, we simulate $n = \sum_{k=1}^{K} n^k$ realisations from the joint distribution, $(\mathbf{X}, \mathrm{Y})$, giving the data matrix and its associated vector of class labels, $(\boldsymbol{X}, \boldsymbol{y})$. We take $\mathbf{X} \sim F_{\mathbf{X}}$ as a mixture of Gaussians i.e., $f_{\mathbf{X}}(\boldsymbol{x}) = \sum_{k=1}^{K} f_{\mathbf{X}|\mathrm{Y}=k}(\boldsymbol{x} \mid \mathrm{Y} = k)\pi^k$, where:

$$\mathbf{X} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{R}\boldsymbol{\mu}^k, \boldsymbol{R}\boldsymbol{\Sigma}^k\boldsymbol{R}^\top\right),$$

and fix $\pi^k = \frac{1}{K}$. $\boldsymbol{R}$ is a $p \times p$ matrix which is included to invalidate CC independence (taking $\boldsymbol{R} = \boldsymbol{I}_p$ would imply generating data in compliance with the NB assumption since $\boldsymbol{\Sigma}^k$ is diagonal). This matrix is populated with realisations of independent $\mathrm{N}(0, 1)$ random variables. It is noted that matrices with entries consisting of standard Normal realisations have an effect similar to applying a rotation to data (i.e., $\boldsymbol{R}$ is close to being orthogonal and having

determinant equal to 1). The mean vector, $\boldsymbol{\mu}^k$, is as in (1.4.1). Independent Uniform $\left(-\frac{C}{p}, \frac{C}{p}\right)$ realisations populate this vector. The parameter C is taken as non-negative and it controls the relative proximity or overlap of the components of the mixture. The covariance matrix, $\boldsymbol{\Sigma}^k$, is diagonal and is as in (1.3.1). Independent Exponential (1) realisations populate its diagonal entries (encouraging elongated Gaussians – adding to the difficulty of the classification task). In light of Projected NB, this design setup is appropriate. Looking at:

$$\mathbf{X}^\top \boldsymbol{V} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{V}^\top \boldsymbol{R}\boldsymbol{\mu}^k, \boldsymbol{V}^\top \boldsymbol{R}\boldsymbol{\Sigma}^k \boldsymbol{R}^\top \boldsymbol{V}\right),$$

we see that should $\boldsymbol{V}^\top = \boldsymbol{R}^{-1}$, the components of $\mathbf{X}$ would be CC independent, and assumption (3.2.1) would be met exactly. The described mixture design will also be used to generate non-Gaussian data. This is done by simulating realisations according to that outlined, and then squaring each entry in the data matrix, $\boldsymbol{X}$. The motivation for doing so is to compare the performance of PGNB and PKNB on Gaussian and non-Guassian data. We expect PKNB's higher degree of flexibility to be adaptable to non-Gaussian data, and therefore to translate into a comparative advantage over PGNB on said data.

Figures 4.1 and 4.2 visualise the Gaussian mixture and squared Gaussian mixture setup, respectively. The top left panel of Figure 4.1 showcases a bivariate Gaussian mixture, and that of Figure 4.2 gives a bivariate Gaussian mixture where the generated data points have been squared (each figure being generated according to the indicated conditions). The top right panels of each figure illustrates the LDA representation (according to the first two LDA components) and the bottom left panels show the representation according to the first two principal components. The bottom right panels of Figure 4.1 and 4.2 provide the representation due to our methods i.e., PGNB and PKNB, respectively (where we have taken $p' = 2$). Comparing the Projected NB representations to that due to LDA and PCA, the most striking difference is the "independence" outcome of PGNB and PKNB. Uniquely among those considered, our projection induces class conditional independence. This observation follows the discussion in Section 3.4 of the connection between Projected NB and independent component analysis.

For the data underlying Figure 4.1, the Bayes error rate is 0.054 and the total Kullback-Leibler divergence (see below) is 1528. SGNB, SKNB, PGNB and PKNB delivered error rates of 0.210, 0.218, 0.056 and 0.081, respectively. On the squared Gaussian mixture data of Figure 4.2, SGNB, SKNB, PGNB and PKNB delivered error rates of 0.144, 0.160, 0.067 and 0.066, respectively. From this we see PGNB to be the best performing model on the Guassian data, while on the non-Gaussian data, PKNB delivers the smallest error rate.

Measuring Difficulty

Objectively distinguishing the difficulty of one classification task from that of another is a concern. In our setup, even when controlling for the number of predictors (p) and classes (K), and for the size of C (the parameter controlling the relative proximity of the components of the mixture), this still remains a challenge. The Bayes error rate for a particular data-set, $(\boldsymbol{X}, \boldsymbol{y})$, is computed via (4.1.1), by taking:

$$\hat{y}_i = \underset{k=1,2,\ldots,K}{\arg\max}\ \mathrm{P}(\mathrm{Y} = k \mid \boldsymbol{x}_i),$$

where P denotes the true conditional probability. This delivers the lowest possible expected error rate obtainable by any classifier and so is a sensible indication of the difficulty of a classification task. Note that computing the Bayes error rate requires exact knowledge of the distributions involved. Also, the above describes an empirical approximation to the true Bayes error rate, $\mathrm{E}\left(\mathbb{1}\left(\hat{\mathrm{Y}}_i \neq \mathrm{Y}_i\right)\right)$.

Another measure of difficulty considered is the Kullback-Leibler divergence. This was discussed in Section 2.2.2 as a quasi-distance measure between two probability distributions. To quantify the relatedness of a collection of K random vectors, say $\{\mathbf{X} \mid \mathrm{Y} = k : k = 1, 2, \ldots, K\}$,

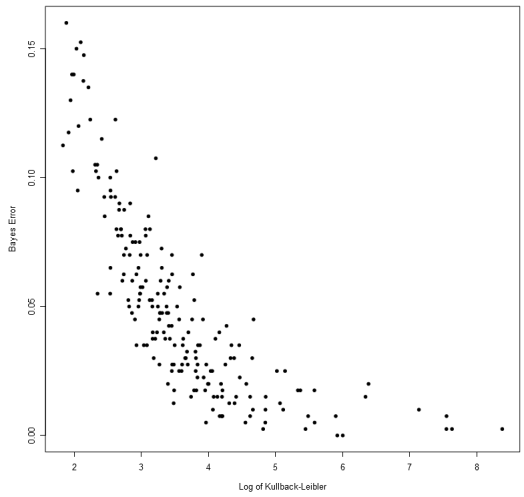


Figure 4.3: Comparison of Bayes error with Kullback-Leibler divergence where $p = 5$, $K = 2$, $C = 3$.

the total sum of the divergence between pairs is used:

$$\begin{aligned} &\sum_{i=1}^{K}\sum_{j=1}^{K}\delta\left(f_{\mathbf{X}|\mathrm{Y}=i}, f_{\mathbf{X}|\mathrm{Y}=j}\right) \\ &= \sum_{i=1}^{K}\sum_{j=1}^{K}\int_{\mathbb{R}^p} f_{\mathbf{X}|\mathrm{Y}=i}(\boldsymbol{x})\,\log\left(\frac{f_{\mathbf{X}|\mathrm{Y}=i}(\boldsymbol{x})}{f_{\mathbf{X}|\mathrm{Y}=j}(\boldsymbol{x})}\right)d\boldsymbol{x}. \end{aligned} \tag{4.2.1}$$

The above can be viewed as a measure of the "class overlap" – that which characterises how difficult it is to distinguish an observation arising from one distribution from that due to another. Remark that, since we are summing over the number of classes, (4.2.1) is only to be used to compare classification difficulty of data-sets having the same K. Also, as with computing the Bayes error rate, (4.2.1) requires exact knowledge of the distributions involved.

Figure 4.3 establishes the inverse relationship between these two measures of classification difficulty. The plot displays the result of a large number of data-sets generated under the indicated conditions. It can be seen that, on average, as the quasi-distance between the distributions increases, the Bayes error rate becomes smaller (i.e., the classification task becomes easier). However, as seen in Figure 4.3, the relationship between the Bayes error rate and the Kullback-Leibler divergence is not one-to-one. Also, it is noted that as p and K gets large, the relationship tends to breaks down, and the interpretation of (4.2.1) as it relates to the Bayes error rate becomes difficult. Nevertheless, the values (4.2.1) are included in Table 4.1.

Results

Tables 4.1 and 4.2 display the results from simulating data according to the Gaussian- and squared Gaussian mixture setup, respectively. The results have been grouped according NB and non-NB type classifiers, and according to the number of classes, K. The tables provide the average error rate, along with its standard deviation, from ten simulations generated under the indicated conditions. In addition, Table 4.1 states the Bayes error rate and total Kullback-Leibler divergence (remarking that the distribution of $\mathbf{X} \mid \mathrm{Y} = k$ where each component of the conditional random vector has been squared is difficult to compute, these two measures of classification difficulty are absent in Table 4.2). Following the description provided at the start of this section, each classification task is defined by the parameters of the components of

Table 4.1: Gaussian mixture data.

Setting	SGNB	SKNB	PGNB	PKNB	CCICA	LDA	QDA	SVM	RF	Difficulty Measure
$K = 2$	0.268	0.268	0.193	**0.192**	0.195	0.285	**0.192**	0.195	0.234	0.190
$p = 2$	(0.021)	(0.021)	(0.015)	(0.017)	(0.016)	(0.031)	(0.016)	(0.017)	(0.019)	*4*
$K = 2$	0.368	0.374	0.146	0.172	0.149	0.332	**0.143**	0.195	0.230	0.140
$p = 5$	(0.026)	(0.021)	(0.015)	(0.064)	(0.011)	(0.029)	(0.014)	(0.019)	(0.020)	*10*
$K = 2$	0.296	0.308	0.072	0.128	0.054	0.183	**0.049**	0.113	0.156	0.049
$p = 10$	(0.023)	(0.024)	(0.009)	(0.013)	(0.009)	(0.021)	(0.007)	(0.012)	(0.014)	*19*
$K = 2$	0.206	0.214	0.040	0.067	0.004	0.371	**0.003**	0.051	0.113	0.003
$p = 25$	(0.030)	(0.030)	(0.009)	(0.016)	(0.004)	(0.021)	(0.003)	(0.012)	(0.021)	*74*
$K = 5$	0.480	0.486	0.468	0.481	**0.462**	0.546	0.468	0.468	0.523	0.462
$p = 2$	(0.018)	(0.018)	(0.020)	(0.014)	(0.023)	(0.015)	(0.018)	(0.016)	(0.021)	*111*
$K = 5$	0.538	0.543	0.266	0.386	0.272	0.576	**0.265**	0.418	0.439	0.264
$p = 5$	(0.017)	(0.019)	(0.018)	(0.014)	(0.018)	(0.019)	(0.017)	(0.016)	(0.016)	*246*
$K = 5$	0.568	0.574	0.128	0.198	0.118	0.648	**0.111**	0.328	0.399	0.109
$p = 10$	(0.013)	(0.014)	(0.010)	(0.009)	(0.012)	(0.009)	(0.012)	(0.016)	(0.010)	*537*
$K = 5$	0.533	0.549	0.055	0.107	0.019	0.677	**0.016**	0.187	0.303	0.014
$p = 25$	(0.012)	(0.011)	(0.029)	(0.002)	(0.012)	(0.003)	(0.013)	(0.015)	(0.003)	*1417*
$K = 10$	0.690	0.693	0.556	0.585	0.557	0.646	**0.555**	0.568	0.650	0.555
$p = 2$	(0.008)	(0.011)	(0.009)	(0.027)	(0.009)	(0.008)	(0.008)	(0.010)	(0.013)	*816*
$K = 10$	0.625	0.631	**0.408**	0.471	0.417	0.706	0.410	0.466	0.501	0.408
$p = 5$	(0.009)	(0.008)	(0.011)	(0.074)	(0.013)	(0.012)	(0.011)	(0.011)	(0.009)	*1907*
$K = 10$	0.753	0.757	0.308	0.397	0.308	0.806	**0.296**	0.542	0.602	0.292
$p = 10$	(0.010)	(0.011)	(0.017)	(0.060)	(0.011)	(0.008)	(0.013)	(0.016)	(0.016)	*10069*
$K = 10$	0.721	0.732	0.132	0.336	0.066	0.822	**0.053**	0.378	0.495	0.046
$p = 25$	(0.008)	(0.010)	(0.015)	(0.157)	(0.006)	(0.015)	(0.005)	(0.007)	(0.012)	*14083*

the mixture:

$$\mathbf{X} \mid \mathrm{Y} = k \sim \mathrm{N}\left(\boldsymbol{R}\boldsymbol{\mu}^k, \boldsymbol{R}\boldsymbol{\Sigma}^k\boldsymbol{R}^\top\right).$$

Once the quantities $\{\boldsymbol{\mu}^k, \boldsymbol{\Sigma}^k : k = 1, 2, \ldots, K\}$ and $\boldsymbol{R}$ are simulated, $n^k = 1000$ realisations from each class conditional distribution are generated ten times. This defines a collection of independently and identically generated data-sets, say $\{(\boldsymbol{X}_m, \boldsymbol{y}_m) : m = 1, 2, \ldots, 10\}$. Each data-set is then randomly segregated into a training-set (80%) and a test-set (20%). The classification error rate, (4.1.1), is computed on the test-set and it is this error rate which is given in Tables 4.1 and 4.2.

In Table 4.1 the choice of C was set equal to 3, and with Table 4.2 we took $C = 10$. The fourth and fifth columns of the tables list the results of PGNB and PKNB, respectively. Projected NB is considered a significant improvement over NB if a standard deviation away from its average error rate does not fall within a standard deviation of the average error of NB. In Table 4.1, we see that PGNB and PKNB have significantly outperformed their standard NB counterparts (i.e., SGNB and SKNB, respectively) in all but a single instance. Inspecting the results of the squared Gaussian mixture, Projected NB is a significant improvement in all but two instances. We also note the extent of the improvement in both tables, with the projection not only improving classification error, but doing so five- to ten-fold in some instances.

Further inspection of Table 4.1 reveals QDA to be the overall best performing classifier (having the lowest, or equal to the lowest, average error rate in all but two instances, and matching the Bayes error rate on three occasions). This is not a surprise given that the Gaussian mixture data is generated exactly according to the assumptions of QDA. The CCICA model performs remarkably well too – QDA is only significantly better (in the sense defined at the start of the preceding paragraph) than CCICA on one occasion. In Table 4.2, the SVM and RF are the best performing classifiers (each being the most accurate classifier on 5 occasions) and CCICA also does well, being competitive with the accuracy of the SVM and the RF. It is noted that in Table 4.1, CCICA does not differ significantly from PGNB and PKNB on the majority of instances. However, on the non-Gaussian data, CCICA significantly outperforms both PGNB and PKNB on most instances.

Our results confirm the findings of John and Langley (1995) regarding the relative performance of SGNB and SKNB on large data-sets. The data underlying Table 4.1 being Gaussian

Table 4.2: Squared Gaussian mixture data.

Setting	SGNB	SKNB	PGNB	PKNB	CCICA	LDA	QDA	SVM	RF
$K = 2$	0.126	0.110	0.115	0.107	0.106	0.157	0.112	**0.103**	0.115
$p = 2$	(0.015)	(0.019)	(0.025)	(0.020)	(0.022)	(0.024)	(0.020)	(0.026)	(0.021)
$K = 2$	0.385	0.316	0.219	0.233	0.221	0.247	0.268	0.220	**0.203**
$p = 5$	(0.021)	(0.027)	(0.023)	(0.030)	(0.022)	(0.035)	(0.019)	(0.024)	(0.033)
$K = 2$	0.290	0.288	0.201	0.240	0.196	0.234	0.226	0.192	**0.182**
$p = 10$	(0.026)	(0.027)	(0.025)	(0.028)	(0.027)	(0.023)	(0.025)	(0.023)	(0.012)
$K = 2$	0.294	0.323	0.226	0.279	0.227	0.262	0.223	**0.222**	0.256
$p = 25$	(0.033)	(0.018)	(0.024)	(0.032)	(0.022)	(0.019)	(0.032)	(0.028)	(0.020)
$K = 5$	0.304	0.287	0.254	0.252	**0.218**	0.333	0.252	0.227	0.231
$p = 2$	(0.015)	(0.010)	(0.008)	(0.024)	(0.017)	(0.016)	(0.009)	(0.010)	(0.016)
$K = 5$	0.368	0.287	0.238	0.256	0.232	0.306	0.287	0.219	**0.210**
$p = 5$	(0.012)	(0.012)	(0.012)	(0.015)	(0.013)	(0.015)	(0.011)	(0.011)	(0.018)
$K = 5$	0.492	0.448	0.349	0.411	0.317	0.396	0.365	**0.281**	0.297
$p = 10$	(0.015)	(0.014)	(0.015)	(0.022)	(0.018)	(0.019)	(0.017)	(0.015)	(0.015)
$K = 5$	0.580	0.602	0.524	0.595	0.504	0.548	0.499	**0.492**	0.510
$p = 25$	(0.011)	(0.021)	(0.016)	(0.019)	(0.021)	(0.010)	(0.009)	(0.010)	(0.014)
$K = 10$	0.691	0.670	0.456	0.516	**0.408**	0.506	0.442	0.596	0.460
$p = 2$	(0.008)	(0.008)	(0.012)	(0.009)	(0.009)	(0.012)	(0.013)	(0.012)	(0.010)
$K = 10$	0.583	0.563	0.524	0.528	0.469	0.548	0.503	0.466	**0.433**
$p = 5$	(0.011)	(0.013)	(0.009)	(0.020)	(0.012)	(0.015)	(0.014)	(0.013)	(0.009)
$K = 10$	0.668	0.655	0.619	0.624	0.528	0.625	0.560	0.505	**0.495**
$p = 10$	(0.012)	(0.014)	(0.013)	(0.011)	(0.011)	(0.011)	(0.010)	(0.012)	(0.014)
$K = 10$	0.740	0.756	0.721	0.744	0.674	0.705	0.665	**0.648**	0.674
$p = 25$	(0.015)	(0.008)	(0.012)	(0.014)	(0.011)	(0.013)	(0.015)	(0.014)	(0.013)

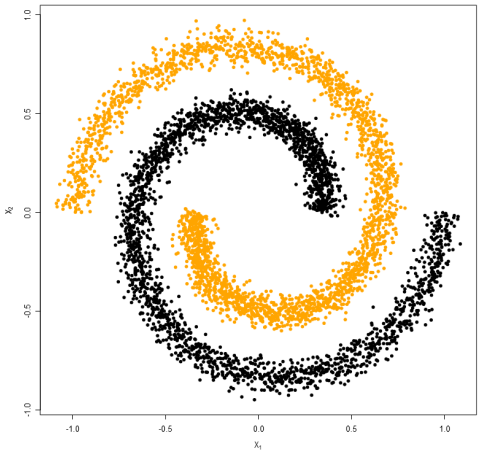
suggests SGNB should enjoy a comparative advantage over the kernel-based counterpart. However, there is no significant difference in the average error between the two classifiers. This indicates that KDE learns the Gaussian structure of the data. On the other hand, with the non-Gaussian data, SKNB performs significantly better than, or equal to, SGNB in all the instances.

4.2.2 "mlbench"

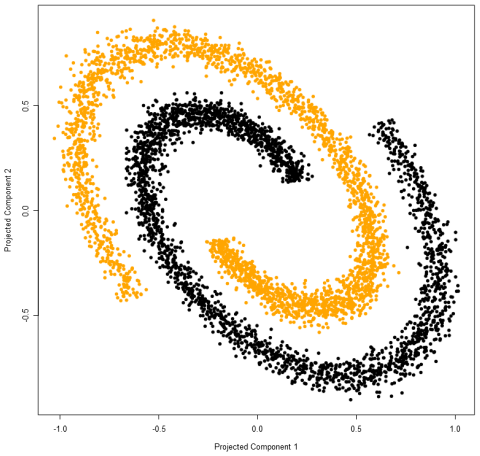
This section relies on the R package "mlbench" (Leisch & Dimitriadou, 2015) for two benchmark classification problems. From the available collection, the two data-sets that have been selected are listed below.

- *Spirals*: A two-class classification problem ($K = 2$) with two predictors ($p = 2$) following a Fibonacci-type spiral.
- *XOR*: A classic nonlinear two-class classification problem ($K = 2$) with inputs uniformly distributed in square structures. The pairs of opposite corners form a class.

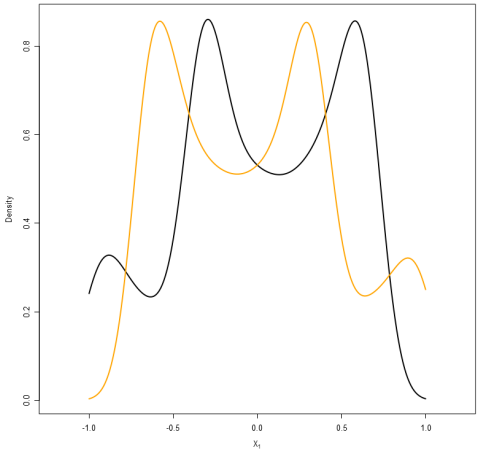
Equally sized training- and test-sets of 5000 observations each were simulated. The result of applying Projected NB, together with those competitors of Section 4.1.1, to Spirals and XOR are given in Table 4.3. It is seen that PKNB delivers a significant improvement over SKNB on Spirals, and that both PGNB and PKNB significantly improve their NB counterparts on XOR. The top left panels of Figures 4.4 and 4.5 visualise, respectively, Spirals and XOR. The corresponding plots to the right in each of the figures demonstrate the rotational impact of the projection matrix. The failure of standard NB on XOR can be explained by studying the second row of plots in Figure 4.5. This reveals the class conditional densities of the original data to be indistinguishable from one-another. The benefit of the projection is seen in the two plots of the third row of this figure. With the first projected component, one class has a unimodal distribution, while the other is distinctly bimodal. Looking at the second projected component, the reverse holds. This means the classes can now be discriminated according



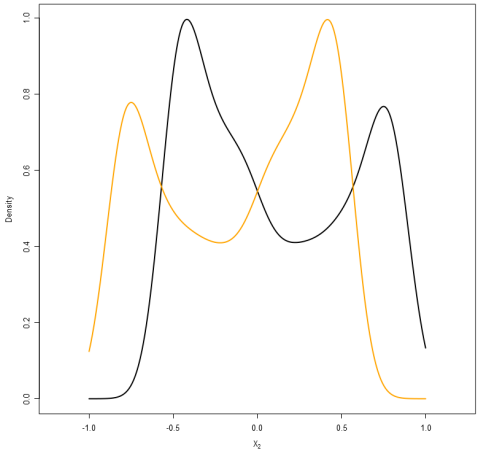
(a) Spirals original representation.



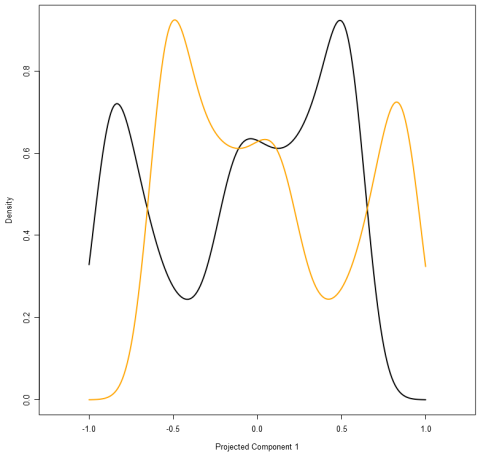
(b) Spirals PKNB representation.



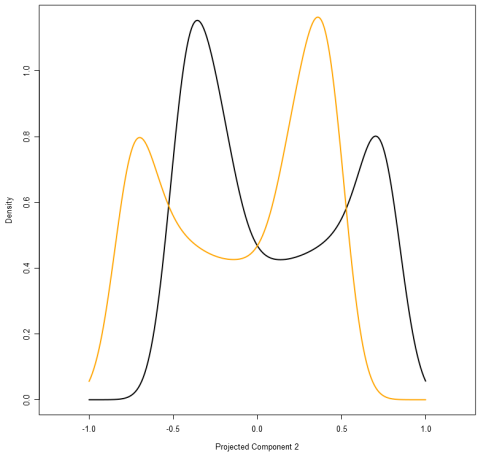
(c) KDE: variable 1.



(d) KDE: variable 2.

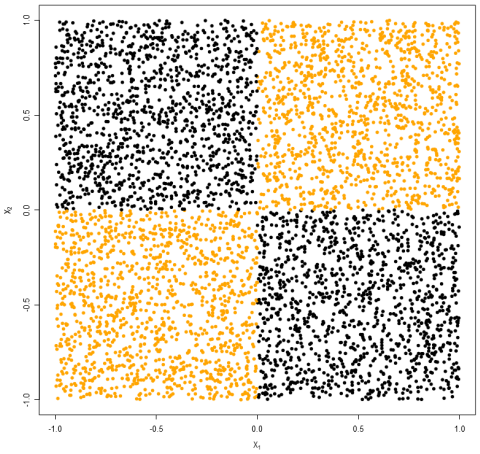


(e) KDE: projected component 1.

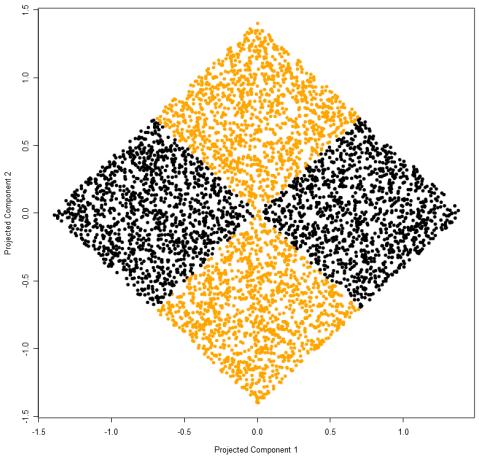


(f) KDE: projected component 2.

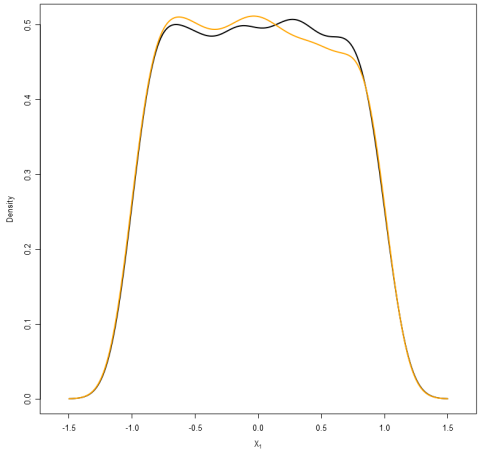
Figure 4.4: Spirals illustration: original- and projected data representations, and kernel density estimates of both original- and transformed variables.



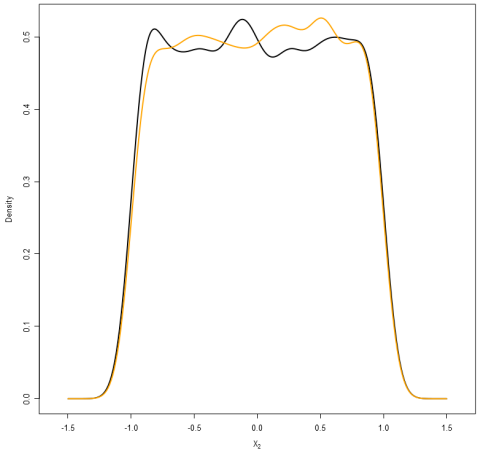
(a) XOR original representation.



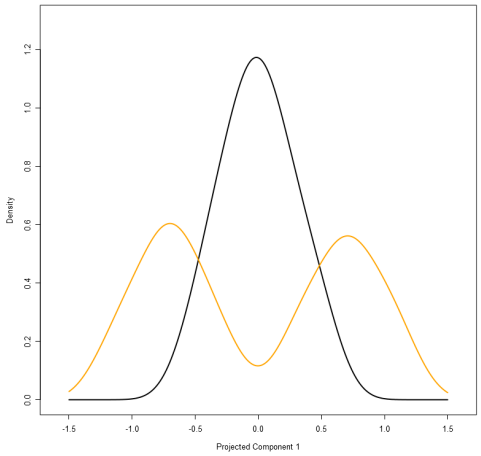
(b) XOR PKNB representation.



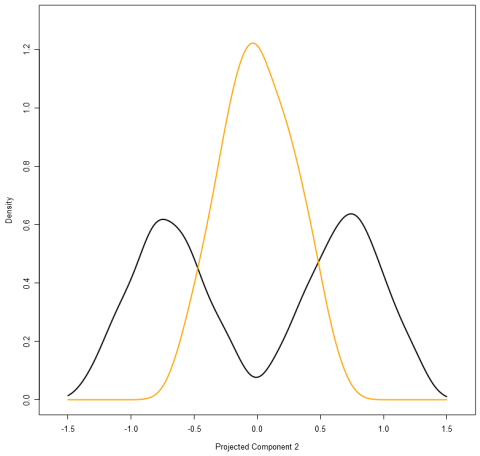
(c) KDE: variable 1.



(d) KDE: variable 2.



(e) KDE: projected component 1.



(f) KDE: projected component 2.

Figure 4.5: XOR illustration: original- and projected data representations, and kernel density estimates of both original- and transformed variables.

Table 4.3: *mlbench* data.

Data-set	SGNB	SKNB	PGNB	PKNB	CCICA	LDA	QDA	SVM	RF
Spirals	0.498 (0.001)	0.067 (0.003)	0.500 (0.001)	0.006 (0.001)	0.069 (0.002)	0.500 (0.001)	0.500 (0.001)	**0.001** (0.000)	0.003 (0.002)
XOR	0.516 (0.053)	0.503 (0.019)	0.011 (0.003)	0.013 (0.007)	0.011 (0.003)	0.519 (0.070)	0.012 (0.004)	0.008 (0.003)	**0.000** (0.000)

to density functions, which explains the success of NB on the projected data. Performing a similar inspection on the second- and third row of plots belonging to the Spirals data-set accounts for why the projection offers a smaller classification accuracy improvement: class discrimination via density functions of the projected data is only marginally better than that of the original data.

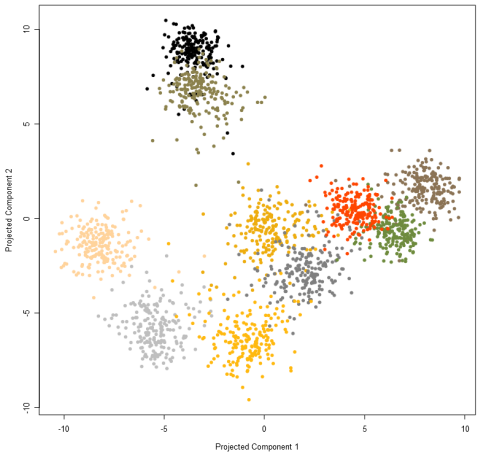
4.3 Real-World Data: UCI Machine Learning Repository and OpenML

The UCI Machine Learning Repository (Dua & Graff, 2019) was established in 1987 and hosts a large collection of data-sets in service of the machine learning community. OpenML (Vanschoren *et al.*, 2013) is described as an "inclusive movement to build an open, organised, online ecosystem for machine learning". It too provides users access to a collection of freely available data-sets. Data from both these sources have been made use of in this section.

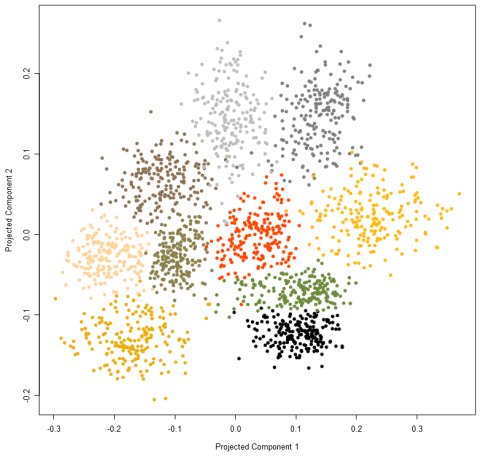
Data Description

The data-sets used are briefly described below.

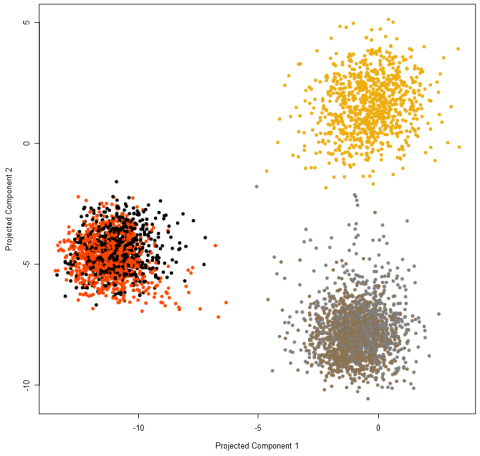
- Banknote: A binary ($K = 2$) classification task related to the identification of forged currency. The data-set has $p = 4$ continuous inputs and contains $n = 1372$ instances.
- Blood Transfusion: A binary ($K = 2$) classification task related to the prediction of whether a patient donated blood during a specific time period. The data-set has $p = 4$ continuous inputs and contains $n = 748$ instances.
- Body-fat: A binary ($K = 2$) classification task related to the identification of obesity. The data-set has $p = 14$ continuous inputs and contains $n = 252$ instances.
- Digits: A ten-class ($K = 10$) classification task related to the identification of handwritten numbers. Originally, the data-set contained 659 inputs (and is called the Multiple Features Data Set). From the available collection, $p = 216$ continuous predictors were extracted. Each class has 200 instances ($n = 2000$).
- Iris: A three-class ($K = 3$) classification problem of iris species due to Sir Ronald Fisher. Inputs ($p = 4$) are continuous measurements. Each class has 50 instances ($n = 150$).
- Letter Recognition: A twenty-six class ($K = 26$) class classification problem of alphabetic characters. Inputs ($p = 16$) are continuous, and the data-set contains $n = 20000$ observations.
- Magic Telescope: A binary ($K = 2$) classification task related to the identification of gamma particles. The data-set of $n = 19020$ observations was constructed using images from a Cherenkov gamma-ray telescope. Inputs ($p = 10$) are continuous.
- Parkinsons: A binary ($K = 2$) classification task related to the identification of Parkinson's disease. Inputs ($p = 22$) are continuous quantities derived from a biomedical voice recordings based on $n = 197$ subjects.



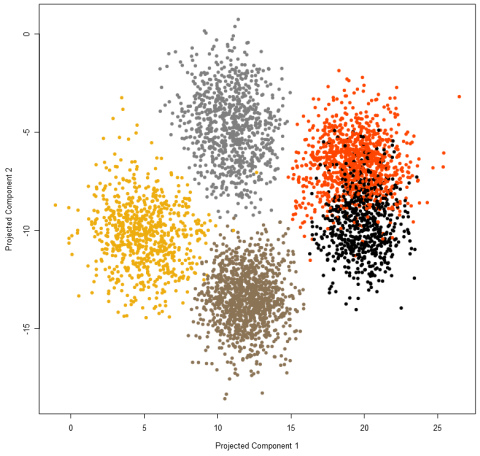
(a) Digits: LDA representation.



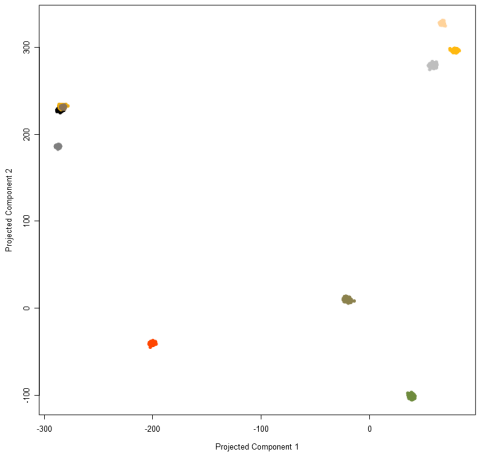
(b) Digits: PGNB representation.



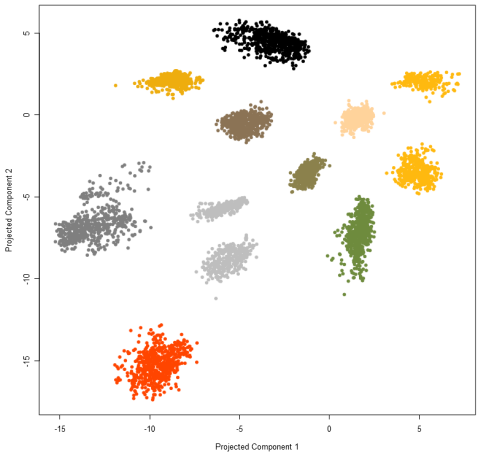
(c) Phoneme: LDA representation.



(d) Phoneme: PGNB representation.



(e) Yale: LDA representation.



(f) Yale: PKNB representation.

Figure 4.6: Visualisation comparison of Projected NB with LDA on selected real-world data-sets.

Table 4.4: UCI and OpenML data.

Data-set	SGNB	SKNB	PGNB	PKNB	CCICA	LDA	QDA	SVM	RF
Banknote	0.161	0.085	0.011	**0.002**	0.003	0.023	0.017	0.007	0.007
	(0.033)	(0.033)	(0.012)	(0.005)	(0.003)	(0.015)	(0.016)	(0.007)	(0.007)
Blood	0.231	0.251	0.222	0.239	**0.219**	0.231	0.227	0.229	0.226
Transfusion	(0.013)	(0.040)	(0.022)	(0.034)	(0.022)	(0.017)	(0.009)	(0.016)	(0.021)
Body-fat	0.206	0.159	0.028	0.044	0.135	0.048	0.139	0.087	**0.004**
	(0.045)	(0.051)	(0.033)	(0.038)	(0.039)	(0.033)	(0.074)	(0.018)	(0.009)
Digits	0.079	0.066	**0.020**	0.260	0.900	0.019	.	0.033	0.032
	(0.008)	(0.013)	(0.006)	(0.022)	(0.014)	(0.009)	.	(0.014)	(0.002)
Iris	0.047	0.040	0.040	0.033	0.033	**0.020**	0.027	0.087	0.047
	(0.018)	(0.028)	(0.028)	(0.024)	(0.041)	(0.018)	(0.015)	(0.069)	(0.018)
Letter	0.359	0.298	0.196	0.157	0.080	0.298	0.114	0.110	**0.035**
Recognition	(0.002)	(0.012)	(0.008)	(0.006)	(0.004)	(0.008)	(0.006)	(0.003)	(0.001)
Magic	0.274	0.237	0.176	0.167	0.182	0.216	0.215	0.195	**0.120**
Telescope	(0.009)	(0.006)	(0.004)	(0.032)	(0.007)	(0.005)	(0.003)	(0.002)	(0.004)
Parkinsons	0.313	0.241	0.144	0.133	0.205	0.128	0.108	0.128	**0.092**
	(0.066)	(0.100)	(0.043)	(0.080)	(0.060)	(0.026)	(0.033)	(0.065)	(0.053)
Phoneme	0.122	0.121	0.086	0.083	0.227	0.074	0.151	0.094	**0.077**
	(0.014)	(0.014)	(0.009)	(0.011)	(0.021)	(0.012)	(0.008)	(0.007)	(0.015)
Seeds	0.105	0.100	0.043	0.057	0.076	**0.038**	0.052	0.152	0.081
	(0.041)	(0.052)	(0.014)	(0.029)	(0.044)	(0.016)	(0.036)	(0.046)	(0.016)
Statlog	0.204	0.179	0.122	0.113	0.139	0.161	0.147	0.136	**0.083**
	(0.007)	(0.007)	(0.010)	(0.020)	(0.013)	(0.011)	(0.010)	(0.012)	(0.008)
Vehicle	0.546	0.375	0.166	0.180	0.177	0.232	**0.150**	0.272	0.249
Silhouettes	(0.049)	(0.024)	(0.023)	(0.024)	(0.040)	(0.026)	(0.024)	(0.042)	(0.013)
Wine	0.028	0.034	**0.011**	0.017	0.056	0.017	**0.011**	0.040	0.023
	(0.020)	(0.024)	(0.015)	(0.016)	(0.044)	(0.025)	(0.015)	(0.048)	(0.024)
Yale	0.000	0.001	0.000	0.000	0.900	0.000	.	0.000	0.000
	(0.000)	(0.001)	(0.000)	(0.000)	(0.007)	(0.000)	.	(0.000)	(0.000)

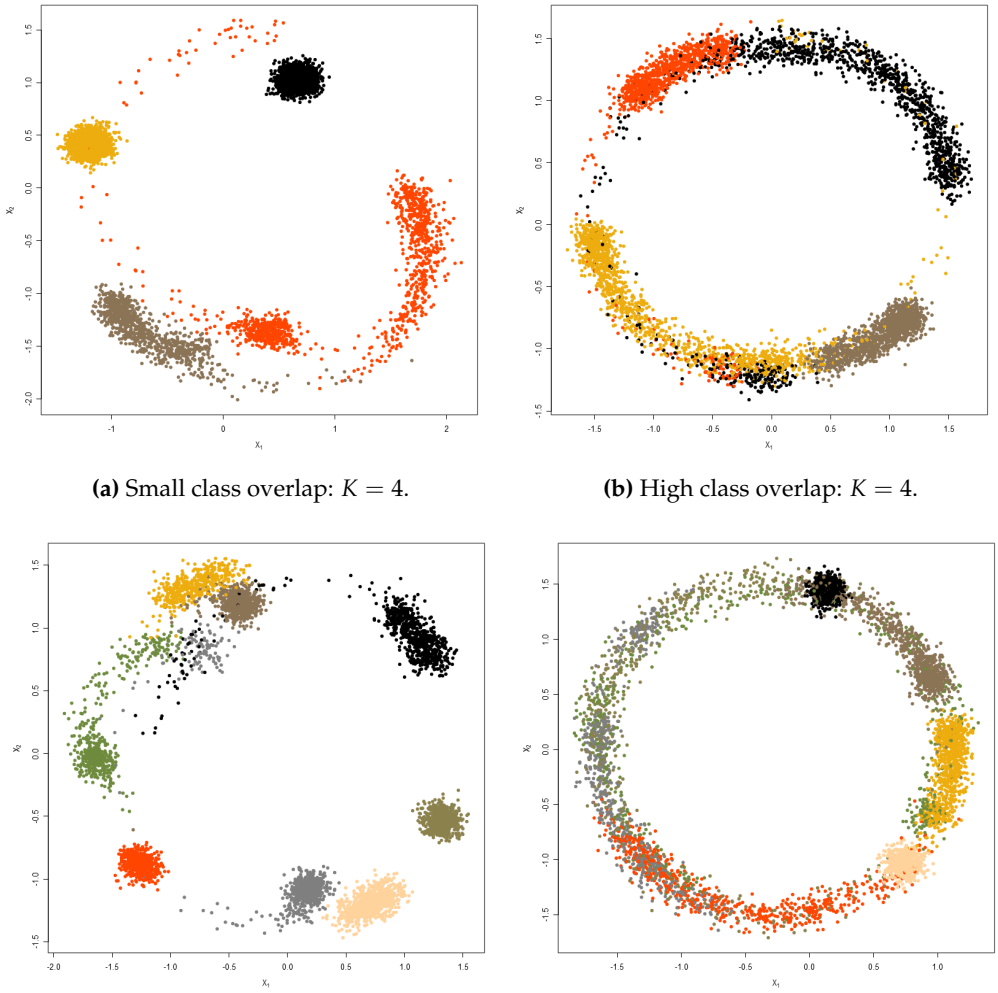
- Phoneme: A binary five-class ($K = 5$) classification task related to distinguishing phonemic sounds. Inputs ($p = 256$) are continuous and there is a total of $n = 5409$ observations.
- Seeds: A three-class ($K = 3$) classification task related to the identification of wheat varieties. Inputs ($p = 7$) are continuous and there is a total of $n = 210$ observations.
- Statlog: A six-class ($K = 6$) classification problem related to the classification of satellite images. Inputs ($p = 36$) are continuous and there is a total of $n = 6435$ observations.
- Vehicle Silhouettes: A four-class ($K = 4$) classification problem related to the prediction of vehicle type. Inputs ($p = 18$) are continuous quantities extracted from a collection of $n = 846$ silhouette images.
- Wine: A three-class ($K = 3$) classification problem related to the distinguishing of different categories of alcoholic beverage. Inputs ($p = 13$) are continuous quantities, and the data-set contains a total of $n = 178$ observations.
- Yale: A ten-class ($K = 10$) classification problem related to the task of facial recognition. Inputs ($p = 1200$) are continuous quantities and there is a total of $n = 5850$ observations.

Results

Table 4.4 provides the five-fold cross-validation results of applying Projected naïve Bayes to real-world data. That is, we randomly partition each data-set into five disjoint folds and use four of these as our training samples. After computing the error rate due to each classifier

on the left-out fold, we repeat the process five times (using each unique combination of four folds as training data and evaluating the classifier on the left out fold) and report the mean and standard deviation. Here, PGNB is a significant improvement over SGNB in all but four of the instances, and PKNB is significantly better than SKNB for eight of the fourteen data-sets presented. Except for the Digits data-set, the instances where the projection does not offer a significant accuracy improvement the performance of SGNB and SKNB are already in line with the other competitors of Section 4.1.1. A comparison of PGNB and PKNB hints at the benefit of a more flexible treatment of the density of the conditional random variable, $\mathbf{X}^\top \boldsymbol{V} \mid \mathrm{Y} = k$. On the real-world data, the kernel approach outperforms the Gaussian counterpart more often than not. Projected NB also significantly improves or matches the accuracy of CCICA for the majority of instances, and is competitive with the accuracy offered by the SVM and RF. Note that the values for the QDA classifier are missing for both the Digits and Yale data-sets. This is due to a high degree of collinearity resulting in fitting issues. Also, we see that on the latter of these data-sets all the methods that were (successfully) fitted gave (near) perfect classification accuracy, except for the CC ICA of Bressan and Vitrià (2002) which gives an error rate matching that due to random guessing. Noting the size of each of the K unmixing matrices this classifier requires, this might be the result of over-parameterisation. On this data-set, due to computational considerations, an exception to the rule-of-thumb given in Section 4.1.2 was made and $p' = 2$ was taken for both PGNB and PKNB.

Figure 4.6 shows Projected NB used as a tool for visualisation. For this purpose, the figure takes $p' = 2$ (regardless of the choice corresponding to the results in Table 4.4, which was according to the rule-of-thumb). As a benchmark for comparison, the corresponding LDA plots (similar to what was given in Figures 4.1 and 4.2) are also provided. It is clear that Projected NB delivers superior visual class discrimination over that of LDA. This is seen in the very small degree of class overlap. What is also clear (but, perhaps not unique to Projected NB in these instances) is the class conditional independence of the projected data. This follows the discussion of the connection between Projected NB and independent component analysis (and, in particular, the role of the second term in the objective function (3.4.2) discussed in Section 3.4).



(a) Small class overlap: $K = 4$.

(b) High class overlap: $K = 4$.

(c) Small class overlap: $K = 8$.

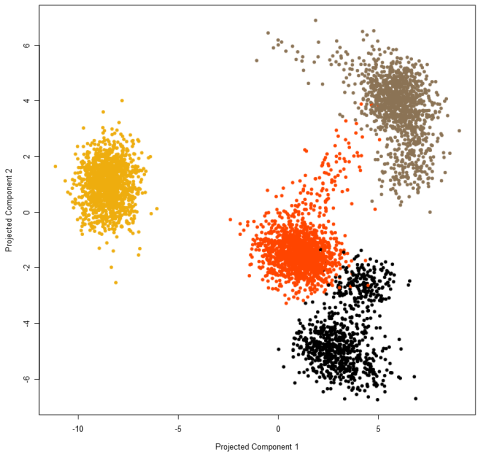
(d) High class overlap: $K = 8$.

Figure 4.7: Illustration: perturbed Gaussian mixtures with $p = 2$.

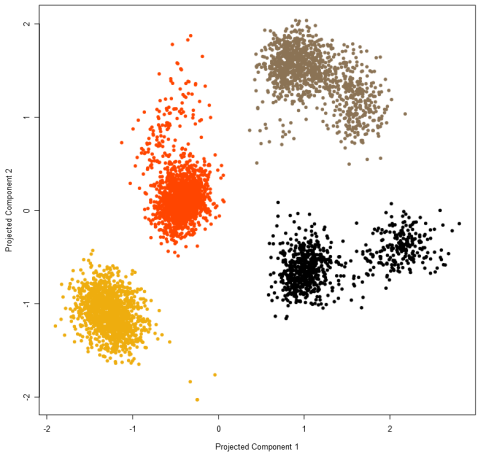
4.4 Visualisation: Perturbed Gaussian Mixtures

This section furthers the demonstration of Projected Naïve Bayes as a tool for visualisation. To this end, perturbed Gaussian mixtures are simulated according to the method described in Hofmeyr (2019b). Figure 4.7 demonstrates these mixtures in two-dimensions (i.e., perturbed bivariate mixtures) under different levels of class overlap for $K = 4$ and $K = 8$, respectively. The perturbation has the effect of inducing non-convexity in the classes (we note, however, that the impact of perturbation in higher dimensions may be different than the vortex-like outcome observed in Figure 4.7). The idea of this section is to examine the visualisation performance of Projected NB when controlling the degree of class overlap and the dimensionality (p) of the simulated data.

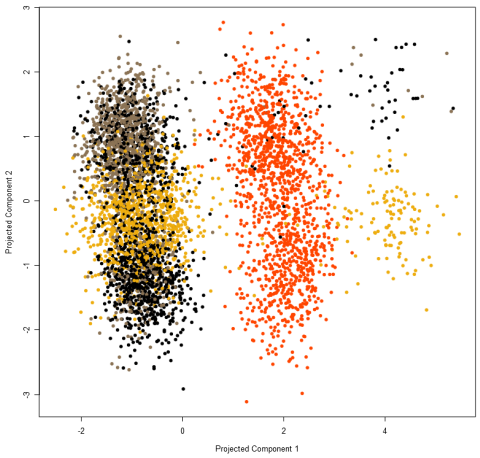
Figures 4.8-4.11 compare the visual success of Projected NB with that of LDA (relying on the first two LDA components) for the task of illustrating the simulated perturbed Gaussian mixtures in two-dimensional space. It is seen that Projected NB offers a visualisation advantage over LDA – particularly as the dimension of the mixtures, p, grows. That is, for $p = 5$, the LDA and Projected NB representations are similar (with Projected NB only offering a modest improvement). However, when taking $p = 15$ or $p = 30$, the LDA representation deteriorates much more seriously than is the case with Projected NB, and it is clear that our model offers



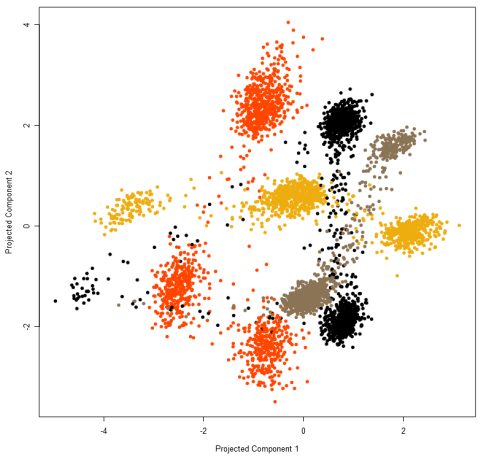
(a) LDA representation: $p = 5$.



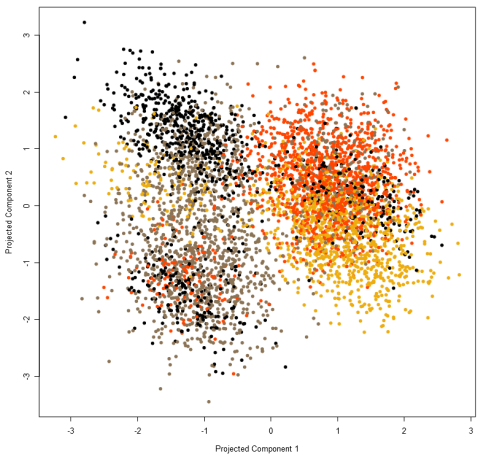
(b) PKNB representation: $p = 5$.



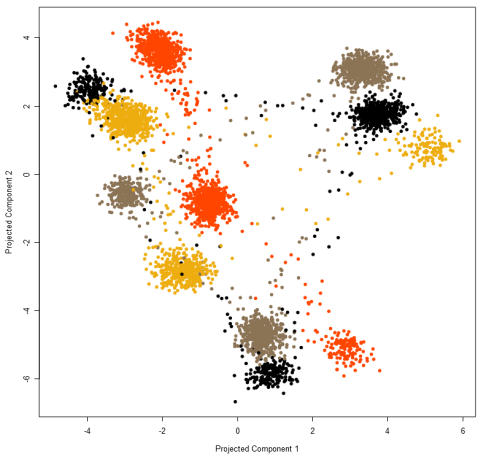
(c) LDA representation: $p = 15$.



(d) PKNB representation: $p = 15$.

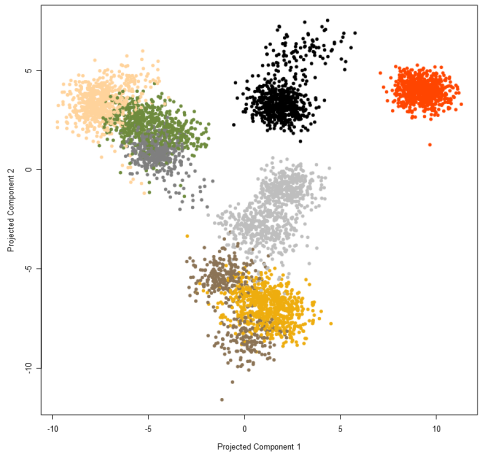


(e) LDA representation: $p = 30$.

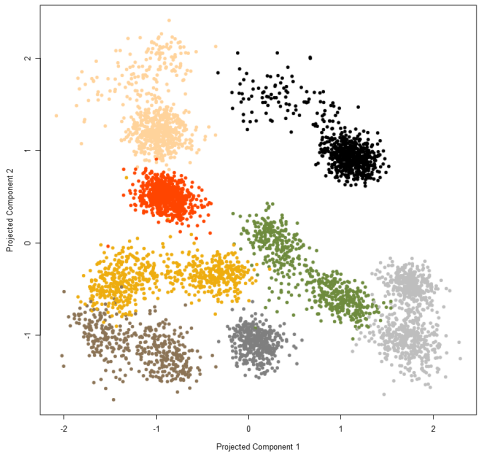


(f) PKNB representation: $p = 30$.

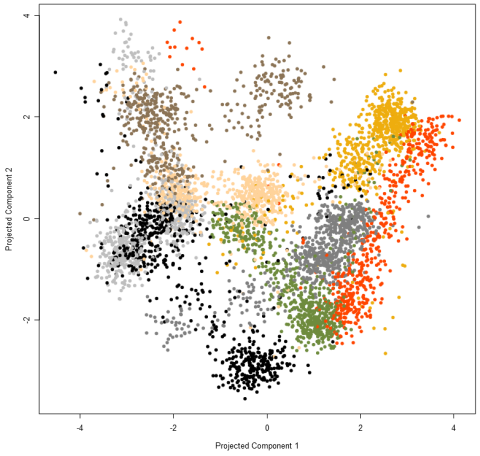
Figure 4.8: Visualisation comparison of LDA and PKNB ($p' = 2$) on perturbed Gaussian mixture with $K = 4$ and small class overlap.



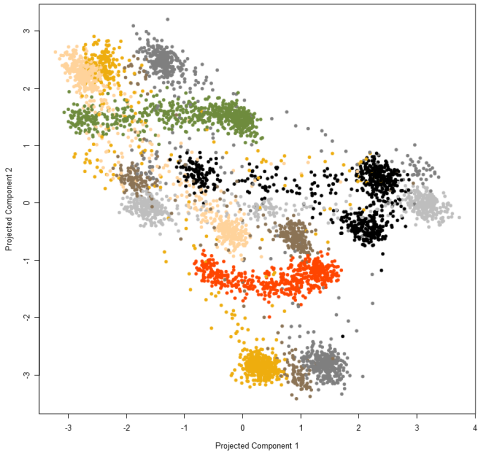
(a) LDA representation: $p = 5$.



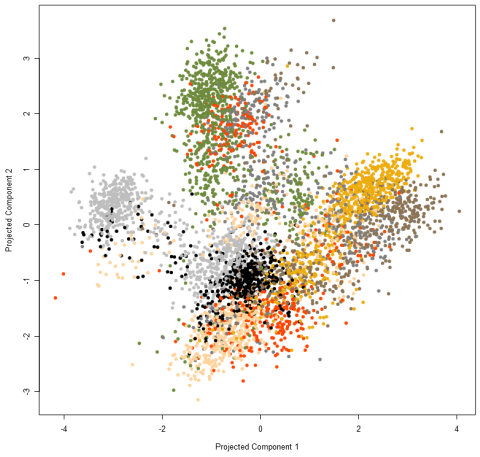
(b) PKNB representation: $p = 5$.



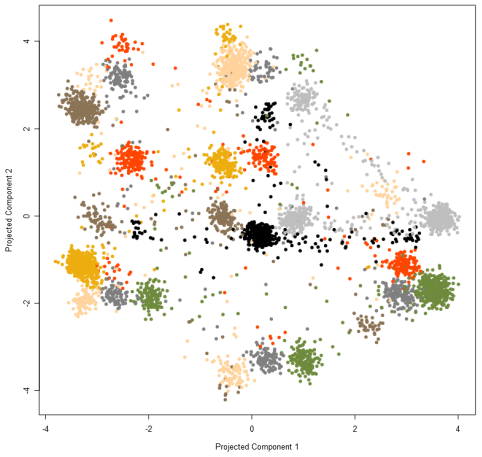
(c) LDA representation: $p = 15$.



(d) PKNB representation: $p = 15$.

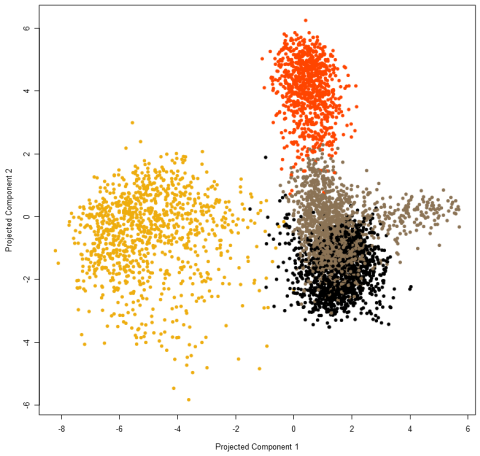


(e) LDA representation: $p = 30$.

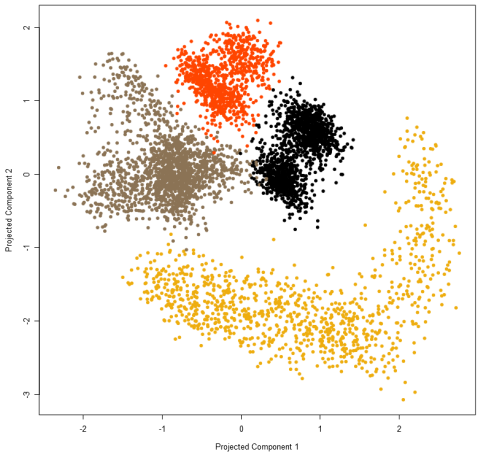


(f) PKNB representation: $p = 30$.

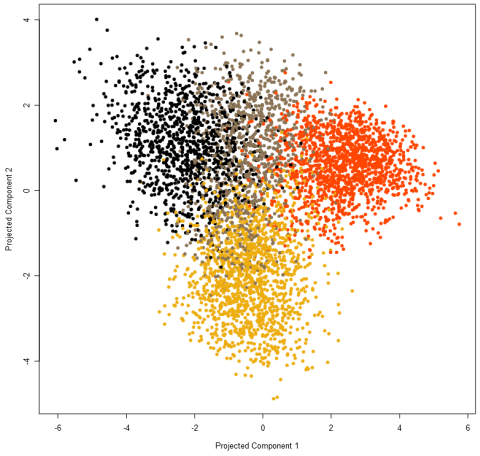
Figure 4.9: Visualisation comparison of LDA and PKNB ($p' = 2$) on perturbed Gaussian mixture with $K = 8$ and small class overlap.



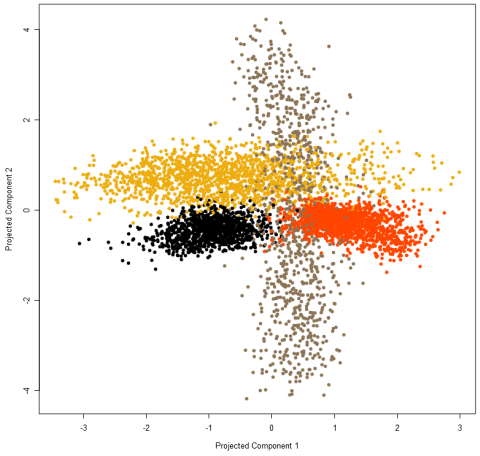
(a) LDA representation: $p = 5$.



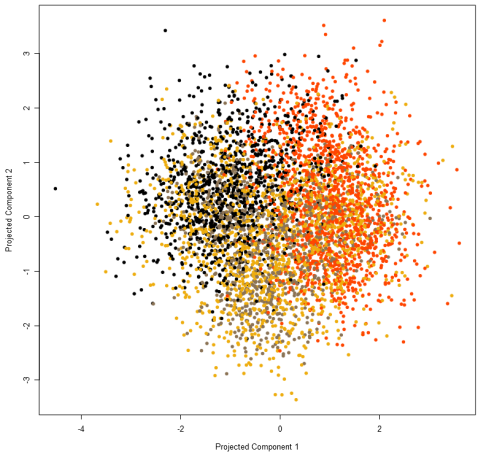
(b) PGNB representation: $p = 5$.



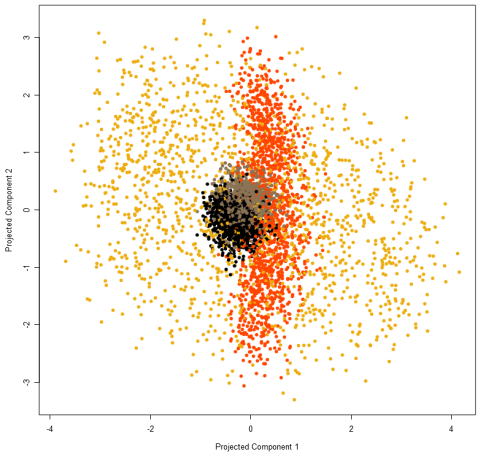
(c) LDA representation: $p = 15$.



(d) PGNB representation: $p = 15$.

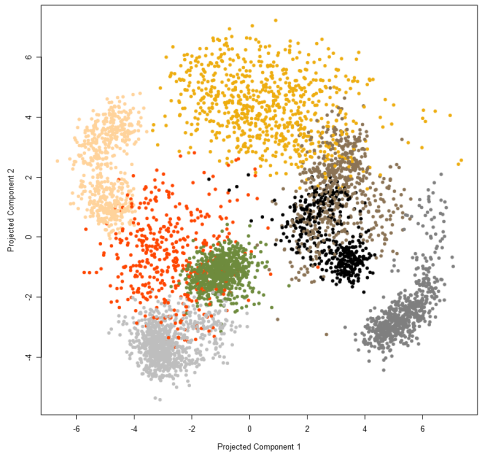


(e) LDA representation: $p = 30$.

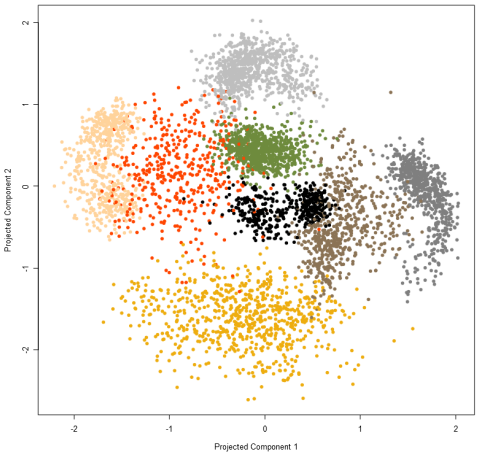


(f) PGNB representation: $p = 30$.

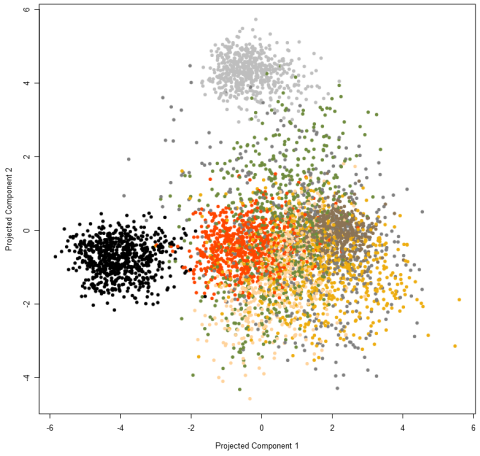
Figure 4.10: Visualisation comparison of LDA and PGNB ($p' = 2$) on perturbed Gaussian mixture with $K = 4$ and high class overlap.



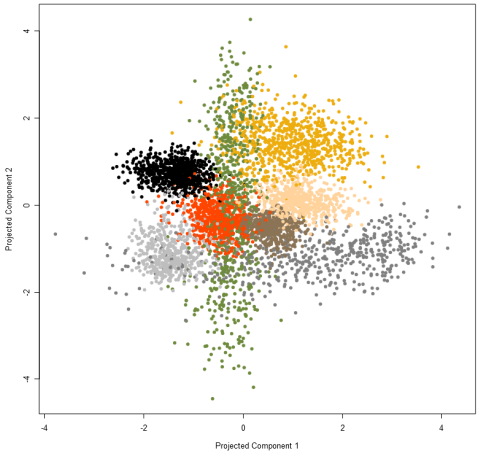
(a) LDA representation: $p = 5$.



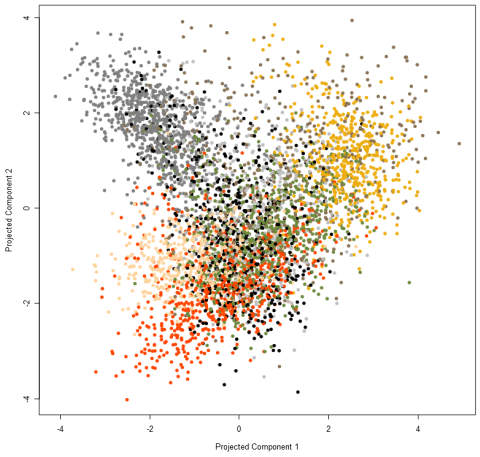
(b) PGNB representation: $p = 5$.



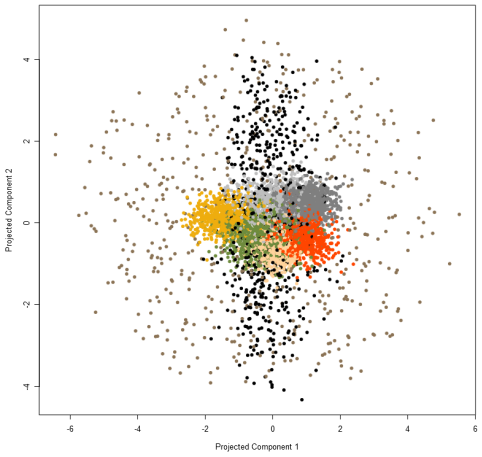
(c) LDA representation: $p = 15$.



(d) PGNB representation: $p = 15$.



(e) LDA representation: $p = 30$.



(f) PGNB representation: $p = 30$.

Figure 4.11: Visualisation comparison of LDA and PGNB ($p' = 2$) on perturbed Gaussian mixture with $K = 8$ and high class overlap.

the superior visualisation. This is seen from the very small degree of class overlap in the projected data. What is also clear, is that our method generally induces a much stronger degree of class conditional independence than does LDA.

4.5 Conclusion

This chapter demonstrated the potential of Projected NB for accurate classification and visualisation purposes. Most importantly, Projected NB was shown as being competitive with popular existing techniques on real-world data from the UCI Machine Learning Repository (Dua & Graff, 2019) and OpenML (Vanschoren *et al.*, 2013). A simulation experiment, designed deliberately in violation of the NB assumption, was also detailed. Projected NB was shown to be significantly better than NB on the majority of instances of both Gaussian and non-Gaussian data, and was also shown to be competitive with other non-linear statistical classifiers. Simulated data from the "mlbench" R package provided insight into the rotational effect of the projection matrix. Finally, Projected NB was shown to be superior to LDA in visualising both real-world data and perturbed Gaussian mixtures.

Chapter 5

Future Work

In this brief chapter we consider two sensible extensions to the method of Chapter 3. Following a discussion of those data-sets to which Projected naïve Bayes can be applied, we propose an amendment to better accommodate data that contain non-continuous inputs. This relies on segregating the collection of predictors according to continuity. Such a separate treatment is presented to combat the degenerating performance of Projected NB when applied to data-sets containing a high proportion of non-continuous inputs. Another extension considered is one that performs automatic variable selection. To this end, inspiration is taken from the version of regularisation implemented by the Sparse-Group LASSO (Simon, Friedman, Hastie & Tibshirani: 2013). Applying regularisation to the rows of the projection matrix delivers a particular type of sparsity which corresponds with variable selection. The potential interpretation and classification accuracy upside of such a projection is discussed, along with the implied optimisation difficulty of the necessary objective reformulation. Finally, a conclusionary overview on this dissertation and Projected naïve Bayes is given.

5.1 Candidate Data-sets

Chapter 3 restricted the scope of Projected naïve Bayes to the case of continuous input features. That is, to assuming that for each of the K classes, the conditional random variable, $\mathrm{X}_j \mid \mathrm{Y} = k$, is continuous for $j = 1, 2, \ldots, p$. Given that we modeled the transformed quantity, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$, as either being Gaussian, (3.2.1), or according to the kernel density estimation of (3.3.1), continuity of the components of $\mathbf{X}$ appears to be a necessary requisite.

However, looking at $\mathbf{X}^\top \boldsymbol{u}_t = \sum_{j=1}^{p} \mathrm{X}_j u_{tj}$, we see that the stochastic variable, $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$, can be thought-of as a (weighted) convolution of the random variables, $\{\mathrm{X}_j \mid \mathrm{Y} = k : j = 1, 2, \ldots, p\}$. Since the convolution of any finite number of discrete- or nominal random variables with even a single continuous random variable delivers a quantity that is continuous in nature, the restriction of all input features having to be continuous can be relaxed to needing only a single input being continuous in order for the modeling of $\mathbf{X}^\top \boldsymbol{u}_t \mid \mathrm{Y} = k$ as a continuous random variable to be valid. (Of course, this is only the case assuming that the weight, say u_{ti}, corresponding to said continuous random variable is non-zero. Also, we have assumed access to standard transformations for the conversion of a nominal variable into numeric discrete variable(s)). This is not to say that we imagine (3.2.1) or (3.3.1) to be a suitable treatment when having many non-continuous inputs among the set, $\{\mathrm{X}_j : j = 1, 2, \ldots, p\}$. It does, however, broaden the class of data-sets to which the proposed method can be applied: those containing some number of discrete- and/or nominal input predictors can be included among the set of candidates.

Amendment: Non-continuous Inputs

Based on our experience, however, the success of Projected NB degenerates with the proportion of non-continuous predictors, X_j. Applying the method to data-sets having a large, or even a moderate, proportion of such inputs delivered classification accuracy worse than standard NB. In addition, it also resulted in optimisation difficulties with the BFGS procedure

often not converging when applied to data where the conditional predictors, $X_j \mid Y = k$, had small variability. For these reasons, it is thought that when faced with a data-set having a large proportion of non-continuous inputs, an alternative approach is likely going to deliver better classification accuracy than when maintaining the Gaussianity of, or the suitability of KDE for, $\mathbf{X}^\top \boldsymbol{u}_t \mid Y = k$.

One such approach would be to segregate the input features into distinct groups, say $\{X_j : j = 1, 2, \ldots, p_1\}$ and $\{X_j : j = p_1 + 1, 2, \ldots, p\}$, according to their continuity. The non-continuous predictors can then be modeled via the methods set out in Section 1.2.2. The likelihood in (3.1.3) can be swapped for the alternative:

$$L(\boldsymbol{V}) = \prod_{i=1}^{n} \frac{\left(\prod_{t=1}^{p'} f_t^{y_i}\left(\boldsymbol{x}_i^\top \boldsymbol{v}_t\right)\right)\left(\prod_{t=p_1+1}^{p} f_t^{y_i}\left(x_{it}\right)\right)\pi^{y_i}}{\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} f_t^{k}\left(\boldsymbol{x}_i^\top \boldsymbol{v}_t\right)\right)\left(\prod_{t=p_1+1}^{p} f_t^{k}\left(x_{it}\right)\right)\pi^{k}}$$

where $p' \leq p_1$. This formulation is a proper treatment of the non-continuous inputs, $\{X_j : j = p_1 + 1, 2, \ldots, p\}$. It allows for the projection, $\boldsymbol{V}$, to be learnt in a way that, although not directly applied to the non-continuous features, still utlises the information offered by them. This is motived by what follows below.

From $\log(\cdot)$ being strictly monotonically increasing, replacing the unknown quantities, f_t^k and π^k, with their estimated counterparts, and discarding those terms that have no bearing on the optimisation task, the (estimated) objective function is given as:

$$\sum_{i=1}^{n}\sum_{t=1}^{p'} \log\left(\hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{v}_t)\right) - \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K}\left(\prod_{t=1}^{p'} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{v}_t)\right)\left(\prod_{t=p_1+1}^{p} \hat{f}_t^{k}\left(x_{it}\right)\right)\hat{\pi}^{k}\right). \tag{5.1.1}$$

This formulation can be compared to that due to Chapter 3. The first term of the above is equivalent to its counterpart in (3.4.2) (although, in the case of (5.1.1), it only relates to a subset of the p inputs). The second term of (5.1.1) captures the influence of the non-continuous inputs on the learning of $\boldsymbol{V}$. Section 3.4 motivates this as an enhancement to the independent component analysis-type first term. Its role is to effect improved classification accuracy. Here, such an effort acknowledges the distinct influence due to continuous- and non-continuous inputs, whereas before this was not the case. That is, we now learn the projection which is conditionally optimal on the estimated probabilities from the non-continuous components. Although the role of the projection becomes less prominent under the above (seeing as it is no longer applied to the full set of predictors), the applicability of Projected NB is substantially extended.

5.2 Sparse Projected Naïve Bayes

It is possible that some of the variables included in $\boldsymbol{X}$ have no relation to the class to which any particular observation belongs. Including such unnecessary variables is expected to worsen classification accuracy. In the case of naïve Bayes, this follows from additional variables implying additional parameters in approximating the posterior probability, $P(Y = k \mid \boldsymbol{x})$. When the parameters in question belong to variables having no predictive power, including these variables will not improve the approximation, but still increases the variance of the estimator. In addition, the inclusion of unnecessary predictors implies a model that is more complicated than it needs to be. The interpretability of a classifier that eliminates irrelevant predictors is also superior to one that does not.

Turning our attention to Projected NB, we notice that variable selection is the case when a particular row of the projection matrix,

$$\boldsymbol{V} = \begin{bmatrix} v_{11} & v_{12} & \dots & v_{1p'} \\ v_{21} & v_{22} & \dots & v_{2p'} \\ \vdots & \vdots & \ddots & \vdots \\ v_{p1} & v_{p2} & \dots & v_{pp'} \end{bmatrix} = \begin{bmatrix} \widetilde{\boldsymbol{v}}_1^\top \\ \widetilde{\boldsymbol{v}}_2^\top \\ \vdots \\ \widetilde{\boldsymbol{v}}_p^\top \end{bmatrix},$$

is zero. That is, the j^{th} input, say X_j, is removed from the model when

$$\widetilde{\boldsymbol{v}}_j^\top = \begin{bmatrix} v_{j1} & v_{j2} & \dots & v_{jp'} \end{bmatrix} = \boldsymbol{0}^\top$$

(i.e., when all the components of the j^{th} row of $\boldsymbol{V}$ are zero). Now, should X_j not have any predictive power, we expect the learning of $\boldsymbol{V}$ according to that maximising an estimate of the likelihood under NB to reflect this, and to deliver the j^{th} row having values that are small in magnitude. However, this learning is unlikely to deliver values that are exactly equal to zero and hence to deliver an interpretational improvement. Also, the aggregation of multiple small contributions from irrelevant predictors might be substantial. This motivates an amendment which encourages those values in $\boldsymbol{V}$ which correspond to unwanted predictors to be put exactly equal to zero.

A matrix with many zero entries is termed "sparse". Following the above discussion, the type of sparsity that delivers variable selection can be achieved by restricting number of non-zero rows of $\boldsymbol{V}$, which can be done by enforcing:

$$\|\boldsymbol{w}\|_0 \leq R \tag{5.2.1}$$

when maximising the likelihood, $L(\boldsymbol{V})$. Here, $\|.\|_0$ is the ℓ_o-norm, and $\boldsymbol{w}$ is the vector of ℓ_2-norms of the rows of $\boldsymbol{V}$:

$$\boldsymbol{w}^\top = \begin{bmatrix} \|\widetilde{\boldsymbol{v}}_1\| & \|\widetilde{\boldsymbol{v}}_2\| & \dots & \|\widetilde{\boldsymbol{v}}_p\| \end{bmatrix}.$$

The non-negative constant, R, controls the number of non-zero rows, and hence the number of predictors, $\{\mathrm{X}_j : j = 1, 2, \dots, p\}$, contributing to the model. Clearly, taking $R \geq p$ will have no impact, and setting $R = 0$ will imply $\boldsymbol{V} = \begin{bmatrix} \boldsymbol{0} & \boldsymbol{0} & \dots & \boldsymbol{0} \end{bmatrix}$ is the sole feasible solution. Noting that restriction (5.2.1) represents an optimisation difficulty, and taking inspiration from the Sparse-Group LASSO (Simon *et al.*, 2013), a convex relaxation of (5.2.1) is obtained by swapping the ℓ_0-norm for the ℓ_1-norm. This gives the alternative restriction when maximising $L(\boldsymbol{V})$:

$$\|\boldsymbol{w}\|_1 = \sum_{j=1}^{p} \|\widetilde{\boldsymbol{v}}_j\| \leq R. \tag{5.2.2}$$

Although (strictly speaking) not directly, (5.2.2) has the effect of encouraging entire rows of $\boldsymbol{V}$ to be exactly equal to zero. The constant, R, now controls the sum of the norms of the rows of the projection. A suitable choice for this value encourages only those rows corresponding to inputs having no predictive power, or to those inputs whose inclusion is expected to worsen the generalizasion error rate, to be set to zero. The problem of maximising an estimate of the likelihood given in (3.1.3) under (5.2.2), and with the columns of $\boldsymbol{V}$ being restricted to having unit norm, is equivalent to the unconstrained maximisation of:

$$\begin{aligned} &\ell(\boldsymbol{U}) - \lambda \sum_{j=1}^{p} \|\widetilde{\boldsymbol{u}}_j\| \\ &= \sum_{i=1}^{n} \log\left(\left(\prod_{t=1}^{p'} \hat{f}_t^{y_i}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^{y_i} \right) - \sum_{i=1}^{n} \log\left(\sum_{k=1}^{K} \left(\prod_{t=1}^{p'} \hat{f}_t^{k}(\boldsymbol{x}_i^\top \boldsymbol{u}_t) \right) \hat{\pi}^{k} \right) - \lambda \sum_{j=1}^{p} \|\widetilde{\boldsymbol{u}}_j\| \end{aligned}$$

with respect to the entries in $\boldsymbol{V}$ and for some $\lambda > 0$. Recall that $\boldsymbol{u}_t = \frac{\boldsymbol{v}_t}{\|\boldsymbol{v}_t\|}$, and define $\widetilde{\boldsymbol{u}}_j^\top = \begin{bmatrix}\widetilde{u}_{j1} & \widetilde{u}_{j2} & \dots & \widetilde{u}_{jp'}\end{bmatrix}$ where $\widetilde{u}_{ji} = \frac{v_{ji}}{\|\boldsymbol{v}_i\|}$. The so-called shrinkage parameter, λ, is inversely related to the constant in (5.2.2), R. In this formulation, the last term can be viewed as a "penalty" that discourages non-sparse solutions. The size of the shrinkage parameter, that which controls the priority of sparsity, can be chosen via cross-validation.

The above is a sensible extension to our NB improvement – automatic variable selection is certainly desirable in many applications. The property of sparsity adds to the parsimonious light in which Projected NB was presented in Section 3.4. It is noted that the sparsity inducing term means that maximisation of the above formulation represents an optimisation challenge that deserves careful attention.

5.3 Conclusion

This dissertation presented gradient-based learning of a projection matrix for improvement of the naïve Bayes classifier. It demonstrated the classification accuracy and visualisation benefit obtained when identifying projections of the data according to a likelihood objection function under the NB assumption. In particular, it was found that a substantial improvement over the classification accuracy of standard NB is possible under an alternative representation of the data. This was seen to be the case on a number of real-world data-sets found in the UCI Repository (Dua & Graff, 2019) and on the OpenML (Vanschoren *et al.*, 2013) website. Projected NB was discussed as extending the flexibility of the standard NB classifier, and as making better use of its degrees of freedom than does standard NB. The proposed method was compared to the application of class conditional independent component analysis to NB due to Bressan and Vitrià (2002). Projected NB was argued as a modified version of CC ICA that preserves its class conditional effect, while being specifically tuned for classification accuracy and being more parsimonious. The proposed method was illustrated to be a sensible tool for visually representing data in low-dimensional space. Our method was shown to give a better visual representation than does linear discriminant analysis on a number of real-world data-sets.

List of References

Anderson, T. W. 1996. R. A. Fisher and Multivariate Analysis. *Statistical Science*, 11(1): 20-34.

Bai, J. & Nie, J. 2004. Using Language Models for Text Classification. *AIRS*.

Boyd, S. & Vandenberghe, L. 2013. *Convex Optimisation*. Cambridge: Cambridge University Press.

Breiman, L. 2002. *Manual On Setting Up, Using, And Understanding Random Forests V3.1* [Online]. Available: https://www.stat.berkeley.edu/breiman/Using_random_forests_V3.1.pdf [2019, October 16].

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. 1984. *Classification and regression trees*. Monterey: Wadsworth, Inc.

Bressan, M. & Vitrià, J. 2002. Improving Naive Bayes Using Class-Conditional ICA, in *IBERAMIA 2002 Proceedings of the 8th Ibero-American Conference on AI: Advances in Artificial Intelligence*. London: Springer-Verlag: 1-10.

Broyden, C. G. 1970. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6: 76-90.

Byatt, D., Coope, I.D. & Price, C.J. 2004. Performance of various BFGS implementations with limited precision second-order information. *The ANZIAM Journal*, 45(4): 511-522.

Cortes, C. & Vapnik, V. 1995. Support-Vector Networks. *Machine Learning*, 20(3): 273-297.

Debnath, L. & Basu, K. 2014. A short history of probability theory and its applications. *International Journal of Mathematical Education in Science and Technology*, 46(1): 1-27.

Dua, D. & Graff, C. 2019. *UCI Machine Learning Repository* [Online]. Available: http://archive.ics.uci.edu/ml [2019, September 12].

El Hindi, K. 2014. Fine tuning the Naïve Bayesian learning algorithm. *AI Communications*, 27(2): 133-141.

Fletcher, R. 1970. A New Approach to Variable Metric Algorithms. *Computer Journal*, 13(3): 317-322.

Goldfarb, D. 1970. A Family of Variable Metric Updates Derived by Variational Means. *Mathematics of Computation*, 24 (109): 23–26.

Hand, D.J. & Yu, K. 2001. Idiot's Bayes—Not So Stupid After All? *International Statistical Review*, 69(3): 385–398.

Hastie, T., Tibshirani, R. & Friedman, J. 2008. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. New York: Springer.

Heidenreich, N., Schindler, A. & Sperlich, S. 2013. Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *Advances in Statistical Analysis*, 97(4): 403-433.

Hoare, Z. 2007. Landscapes of Naïve Bayes classifiers. *Pattern Analysis and Applications*, 11(1): 59-72.

Hofmeyr, D. 2019a. Fast Exact Evaluation of Univariate Kernel Sums. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hofmeyr, D. 2019b. Improving Spectral Clustering using the Asymptotic Value of the Normalised Cut. *Journal of Computational and Graphical Statistics*.

Hyvärinen, A. 1999. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10(3): 626–634.

Hyvärinen, A., Karhunen, J. & Oja, E. 2001. *Independent Component Analysis*. New York: J. Wiley.

James, G., Witten, D., Hastie, T. & Tibshirani, R. 2013. *An Introduction to Statistical Learning: with Applications in R*. New York: Springer.

Jiang, L., Zhang, L., Li, C. & Wu, J. 2019. A Correlation-Based Feature Weighting Filter for Naive Bayes. *IEEE Transactions on Knowledge and Data Engineering*, 31(2): 201–213.

John, G.H. & Langley, P. 1995. Estimating Continuous Distributions in Bayesian Classifiers, in *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann Publishers: 338-345.

Kohavi, R. 1996. Scaling Up the Accuracy of Naive-Bayes Classifiers: A Decision-Tree Hybrid, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. Portland: AAAI Press.

Kohonen, T. 1996. Emergence of invariant-feature detectors in the adaptive-subspace self-organizing map. *Biological Cybernetics*, 75: 281–291.

Langley, P. & Sage, S. 1994. Induction of Selective Bayesian Classifiers, in *Proceedings of the Tenth international conference on Uncertainty in artificial intelligence*. Seattle: Morgan Kaufmann Publishers Inc.: 399-406.

Leisch, F. & Dimitriadou, E. 2015. *Machine Learning Benchmark Problems*. [Online]. Available: https://cran.r-project.org/web/packages/mlbench/mlbench.pdf [2019, September 5].

Maron, M. 1961. Automatic Indexing: An Experimental Inquiry. *Journal of the ACM (JACM)*, 8(3): 404–417.

Metsis, V., Androutsopoulos, I., & Paliouras, G. 2006. Spam Filtering with Naive Bayes - Which Naive Bayes? *CEAS*, 16: 28-69.

Nawi, N. M., Ransing, M. R. & Ransing, R. S. 2006. An Improved Learning Algorithm Based on The Broyden-Fletcher-Goldfarb-Shanno (BFGS) Method For Back Propagation Neural Networks, *Sixth International Conference on Intelligent Systems Design and Applications*:152-157.

Ng, A. 2019. *Support Vector Machines* [Online]. Available: http://cs-229.stanford.edu/notes/cs229-notes3.pdf [2019, September 12].

Palacios-Alonso, M., Brizuela, A. & Sucar, C., 2010. Evolutionary Learning of Dynamic Naive Bayesian Classifiers. *Journal of Automated Reasoning*, 45(1): 21–37.

Papakonstantinou, J. M. 2009. *Historical Development of the BFGS Secant Method and Its Characterization Properties*. Unpublished doctoral dissertation, Rice University.

Rice, J.A. 2007. *Mathematical Statistics and Data Analysis*. Belmont: Duxbury Press.

Rish, I. 2001. *An empirical study of the naive Bayes classifier* [Online]. Available: https://pdfs.semanticscholar.org/3cd1/df035ce7a022eefaf3190627ffcd5e43eca3.pdf?ga=2.127746509.134247-1461.1565937456-1223996292.1559898698 [2019, August 16].

Shanno, D. F. 1970. Conditioning of quasi-Newton methods for function minimization. *Mathematics of Computation*, 24(111): 647–656.

Silverman, B.W., 1986. *Density estimation for statistics and data analysis 1st edition*. London: Chapman and Hall.

Simon, N., Friedman, J., Hastie, T. & Tibshirani, R. 2013. A Sparse-Group Lasso. *Journal of Computational and Graphical Statistics*, 22(2): 231-245.

Stigler, S. M. 1983. Who Discovered Bayes' Theorem? *The American Statistician*, 37(4): 290-296.

Tsymbal, A. & Puuronen, S. 2003. Ensemble feature selection with the simple Bayesian classification. *Information Fusion*, 4(2): 87-100.

Vanschoren, J., Van Rijn, J. N., Bischl, B. & Torgo, L. 2013. OpenML: networked science in machine learning. *SIGKDD Explorations*, 15(2): 49-60.

Vilalta, R. & Rish, I. 2003. A Decomposition of Classes via Clustering to Explain and Improve Naive Bayes, in *Proceedings of the 14th European Conference on Machine Learning*. Cavtat-Dubrovnik: Springer-Verlag: 444-455.

Walker, M. 2017. *Why We Sleep*. New York: Scribner.

Yao, W. & Zhao, Z. 2013. Kernel Density-Based Linear Regression Estimate. *Communications in Statistics – Theory and Methods*, 42(24): 4499–4512.

Zare, D. 2011. *tight bounds on probability of sum of laplace random variables* [Online]. Available: https://mathoverflow.net/questions/66763/tight-bounds-on-probability-of-sum-of-laplace-random-variables [2019, September 13].